Title: 2014 Final Reports from the Los Alamos National Laboratory Computational Physics Student Summer Workshop

Author(s): Runnels, Scott Robert; Ho, Andrew; Garrett, Kerry; Levy, Caleb; Schloss, James; Daligault, Jerome Olivier; Gill, Nathanael Matthew; Heinonen, Robin Alrik; Saumon, Didier; Starrett, Charles Edward; Holgado, Aaron; Holladay, Robert Tyler; Urbatsch, Todd James; Wollaber, Allan Benton; Cleveland, Mathew Allen; Joglekar, Archis; Ortega, Mario; Vold, Erik Lehman; O'Neill, Brian; Magolan, Benjamin; Harrison, Alan Kent; et al.

Intended for: Web

Issued: 2014-09-04

2014 Final Reports

From the

Los Alamos National Laboratory Computational Physics Student Summer Workshop

Assembled by: Scott R. Runnels, Ph.D.
Workshop Coordinator and University Liaison for LANL's Advanced Scientific Computing Program

LA-UR-14-????

Included in this Report

Contents

Introduction

Philosophy of the Workshop

The two primary purposes of LANL's Computational Physics Student Summer Workshop are (1) To educate graduate and exceptional undergraduate students in the challenges and applications of computational physics of interest to LANL, and (2) Entice their interest toward those challenges. Computational physics is emerging as a discipline in its own right, combining expertise in mathematics, physics, and computer science. The mathematical aspects focus on numerical methods for solving equations on the computer as well as developing test problems with analytical solutions. The physics aspects are very broad, ranging from low-temperature material modeling to extremely high temperature plasma physics, radiation transport and neutron transport. The computer science issues are concerned with matching numerical algorithms to emerging architectures and maintaining the quality of extremely large codes built to perform multi-physics calculations. Although graduate programs associated with computational physics are emerging, it is apparent that the pool of U.S. citizens in this multi-disciplinary field is relatively small and is typically not focused on the aspects that are of primary interest to LANL. Furthermore, more structured foundations for LANL interaction with universities in computational physics is needed; historically interactions rely heavily on individuals' personalities and personal contacts. Thus a tertiary purpose of the Summer Workshop is to build an educational network of LANL researchers, university professors, and emerging students to advance the field and LANL's involvement in it.

This was the fourth year for the Summer Workshop and the third in a series of reports [57] [58]. As before, the workshop's goals were achieved by attracting a select group of students recruited from across the U.S. and immersing them for ten weeks in lectures and interesting research projects. The lectures provided an overview of the computational physics topics of interest along with some detailed instruction while the projects gave the students a positive experience accomplishing technical goals. Each team consisted of two students working under one or more LANL mentors on specific research projects associated with predefined topics. This year, the topics were on algorithms for shock hydrodynamics, plasticity, plasma mixing in ICF applications, 3-T plasma physics, new algorithms for GPUs and MICs, a computational study of warm dense matter, mesh generation, verification problems for hydrodynamics, turbulence modeling, and visualization. The students' growth was furthered by their participation on teams where their teammates were sometimes of a different academic year. It also developed their skills by requiring them to produce written and oral reports that they presented to peers, mentors, and management.

Funding and Participation Profile

LANL Staff

The Advanced Scientific Computing (ASC) Program at Los Alamos National Laboratory sponsors the Summer Workshop by funding the workshop coordinator under the University Liaison budget and paying the lease for the workshop facility. Funding for the students' stipends come from a variety of programmatic sources. A large majority of them fall under various projects that are part of the ASC Program, but a few other programs also provide funding for some students. This year, there were eighteen mentors, up slightly from last year. The mentor participation amongst different divisions included XCP, XTD, T, CCS, and EES. This broad participation is welcomed and it is hoped that it continues in future years.

Students

Fifty-two students applied for admission to the workshop, all eligible U.S. citizens with the breakdown shown in the chart that follows. The twenty-two who ultimately were selected and participated were from the following schools: Auburn, Berkeley, Georgia Tech, MIT, Univ. of Maine, Mississippi State, Purdue, St. Olaf College, Stanford, Stony Brook Univ., Texas A&M, UC-San Diego, Univ. Illinois , Univ. Minnesota, Univ. New Mexico, Univ. Washington, Univ. Wyoming, and Virginia Tech. A breakdown of applicants and participants by academic year is also shown in the chart on the next page.

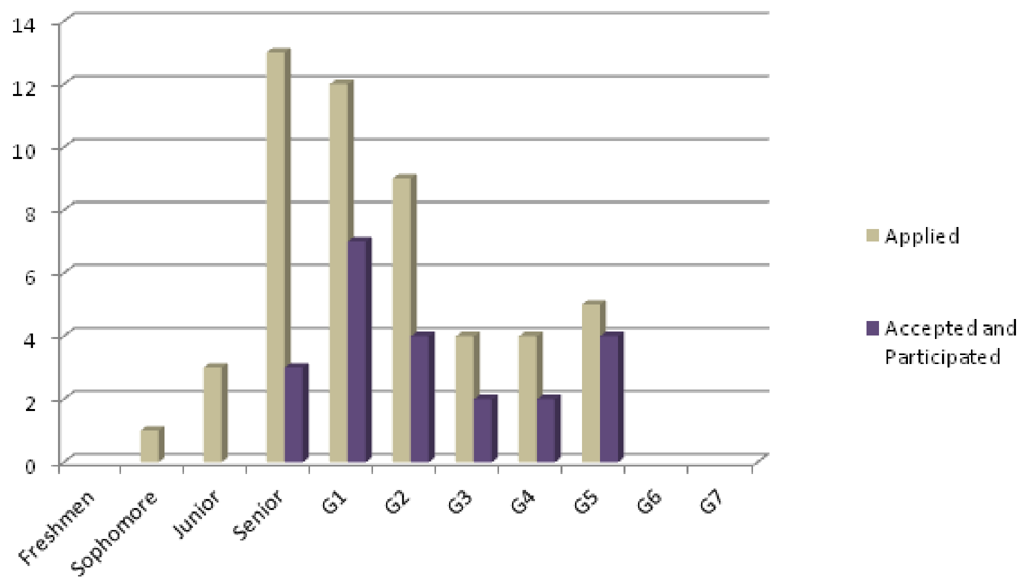


Figure 1: This figure shows the number of students who applied to the Summer Workshop and how many were accepted and participated, broken down by academic year. In this figure "G1" means "first-year graduate student" at the time of the workshop,.i.e., starting their first year of graduate school in the fall after the workshop. "G2" means "second-year graduate student" at the time of the workshop, i.e., starting their second year of graduate school in the fall after the workshop.

Lectures

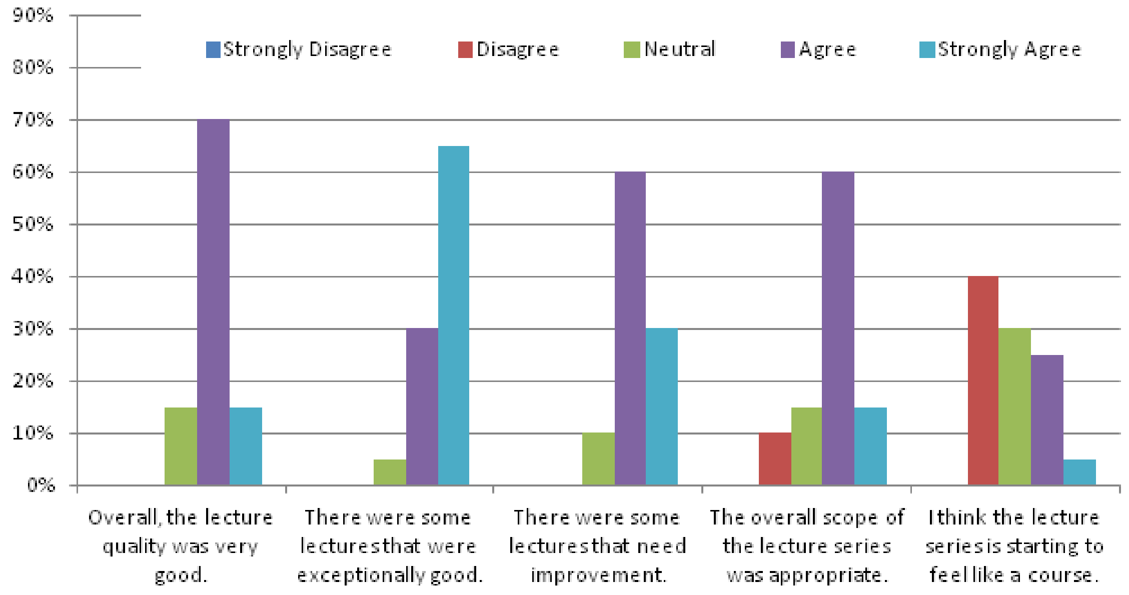
In this fourth year of the Summer Workshop, an effort was made to more tightly integrate the lectures provided to the students. The increased integration is part of a plan to transform the stand-alone lectures into a sequence exhibiting a more course-like feel. A foundational lecture at the beginning of the Summer Workshop, introducing the fundamentals of transport theory, provided a common basis upon which several other lectures could build. Also the development of a one-dimensional hydrocode was performed in class; the resulting code provided a basis for other lecture materials and for exploratory studies that some of the students performed at the beginning stages of their projects. The approximately 35 hours of lectures, for which the students' attendance was required, were augmented with other lectures and demonstrations for which the students' attendance was optional. These lectures were provided to help students who were lacking certain skills develop them quickly to aide them during the summer. The lectures included a tutorial on C++ object-oriented programming, Python programming, and Unix.

The lectures were scheduled to be most frequent in the beginning of the workshop, when the students' research was just getting started and they needed the most background information. Their frequency dropped significantly until there were no lectures at all in the latter weeks of the workshop so that the students could focus on their research. The lectures are summarized in the table that follows.

Title	Hrs.	Lecturer	Notes
Essentials of Transport Equations	2	S. Runnels	Foundational
Introduction to Lagrange Hydro	1	S. Runnels	
Introduction to Hydro Terminology	1	S. Runnels	
Introduction to Artificial Viscosity	3/4	S. Runnels	
Introduction to Python	1	D. Israel	Optional
Tutorial in cpp Programming	2	S. Runnels	Optional
Python for Scientific Computing Applications	1	D. Israel	Optional
Live Demo: Development of a 1-D Gas Hydrocode	1	S. Runnels	
Plasticity Modeling in ASC Hydrocodes	1	S. Runnels	
Live Demo: Adding Plasticity to a 1-D Hydrocode	1	S. Runnels	Optional
Survey of Lagrange Hydro Methods	1	N. Morgan	
Survey of ALE Methods	1	N. Morgan	
Unix Tutorial	1	S. Runnels	Optional
Interface Reconstruction Methods	1	M. Shashkov	
Introduction to Eulerian Hydro	2	W. Rider (Sandia)	
History of Hydrocodes	1	W. Rider (Sandia)	Optional
LaTex Tutorial	1/2	S. Runnels	Optional
Approaching a Complex Physics Problem	1	D. Saumon	
Introduction to Thermal Radiation Transport	1	T. Urbatsch	
Radiation Hydrodynamics	1	S. Ramsey	
An Introduction to Molecular Dynamics	1	C. Starrett	
Particle Transport in Plasmas	1	J. Daligault	
Multi-Material Equilibration Techniques	1	A. Harrison	
Intro to High-Performance Computing at LANL	1	R. Cunningham	Optional
Methods of Software Verification	1	N. Malaya (UT-Austin)	
SQA for Scientific Computing Applications	1	N. Malaya (UT-Austin)	
Monte Carlo Foundations	1	F. Brown	
Introduction to MCNP	1	F. Brown	
Parallel Programming Algorithms	1	R. Robey	
Turbulence Modeling	2	D. Israel	
Plasma Physics Modeling	1	E. Vold	
Particle-Based Methods	1	R. Reisner	
V & V and Uncertainty Quantification	1	J. Kamm & G. Weirs (Sandia)	
Script-Driven Mesh Generation	1	D. Herring	
Mesh Generation Demo	1	D. Herring	Optional

Continuous Improvement

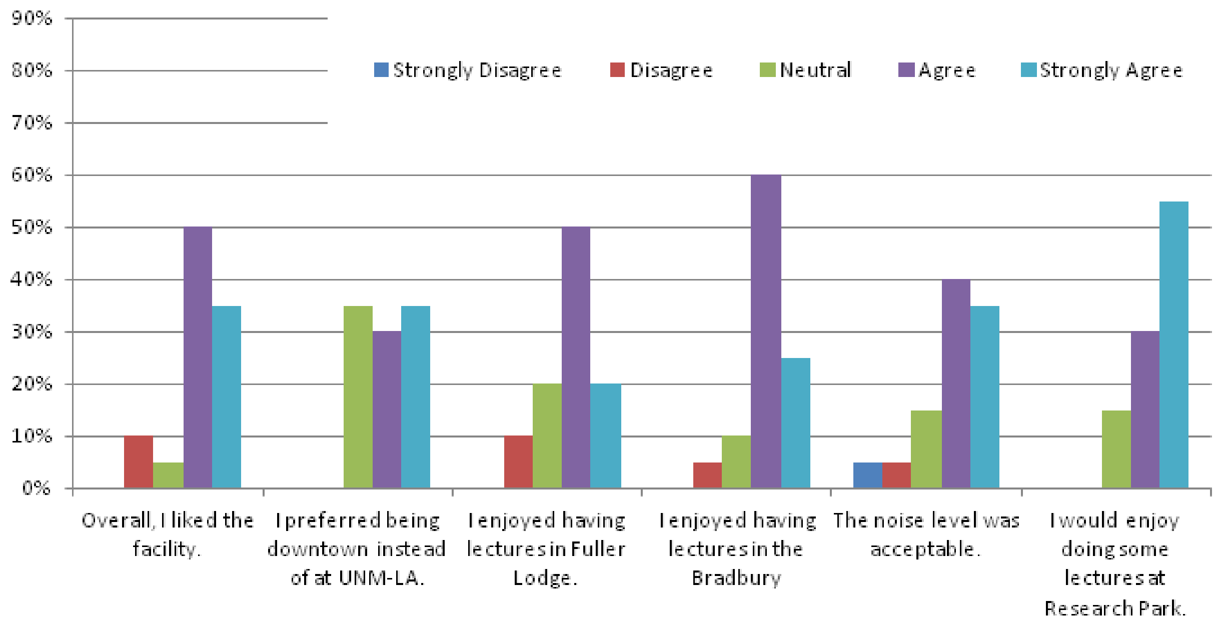
Twenty-one of the twenty-two students participated in a voluntary, anonymous survey as part of the Summer Workshop's ongoing effort to continuously improve and meet its goals. The students responded to a series of questions with the response "Strongly Agree", "Strongly Disagree", "Neutral", "Disagree", or "Strongly Disagree". Although lasts year's responses were very positive, there were issues with the furniture. They were remedied this year by acquiring new, much larger furniture and new chairs. There were also problems with temperature control last year, but the new facility this year (eventually) remedied that. Because of those changes, questions about temperature control and furniture were removed from the survey and were replaced with new ones in order to seek out and expose more areas for improvement. In addition, in an attempt to more strongly coordinate the lecture series and shepherd it toward a course-like feel, the students were surveyed on the progress in that specific respect. The survey results are presented in the figures that follow; below each figure are analyses and summaries of improvements for next year.



Overall Result: Lectures were good, but we need to adjust emphasis/scope and tie them together better.

Analysis: The above survey results indicate that, overall, the quality of the lectures were good but there was a wide range of variability to their quality. The scope of the lectures appear to be acceptable, however on that point, some students disagree. Some disagreement is expected because the students are at different levels, however, comments on individual surveys indicate there was too strong of an emphasis on hydrodynamics. That criticism transcends academic level and does seem legitimate; most of the projects are hydro-related, as well, and that emphasis affects the lectures. Next year, the hydro lectures will be reduced in favor of greater emphasis on the other topics. The students tended to disagree with the assertion that the lectures are beginning to feel like a course. That assertion was placed intentionally, knowing that more progress is required in order to achieve a positive response.

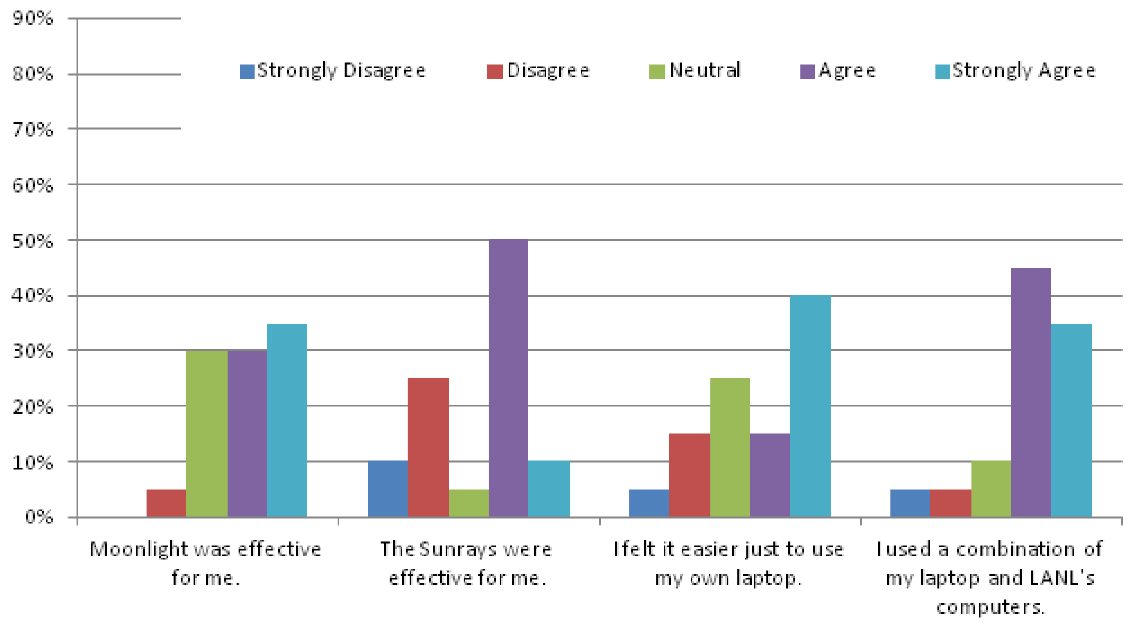
Summary of Improvements for Next Year: Condensing hydro-related lectures in favor of other areas and further developing consistency between lectures, which is required to establish a course-like feel.



Overall Result: The new facility is an improvement but should be adjusted.

Analysis: The above survey results seem to indicate that the new facility for the Summer Workshop is acceptable and is an improvement over previous years'. Most enjoyed lectures at Fuller Lodge and the Bradbury Science Museum, however use of those facilities for lectures will be reduced next year in favor of Research Park, which provides more modern and standard meeting rooms. There is some disapproval with the facility, especially with regard to the noise (conversation) level. Other comments indicate that it will be beneficial to add another meeting room and spread out some of the cubicles next year.

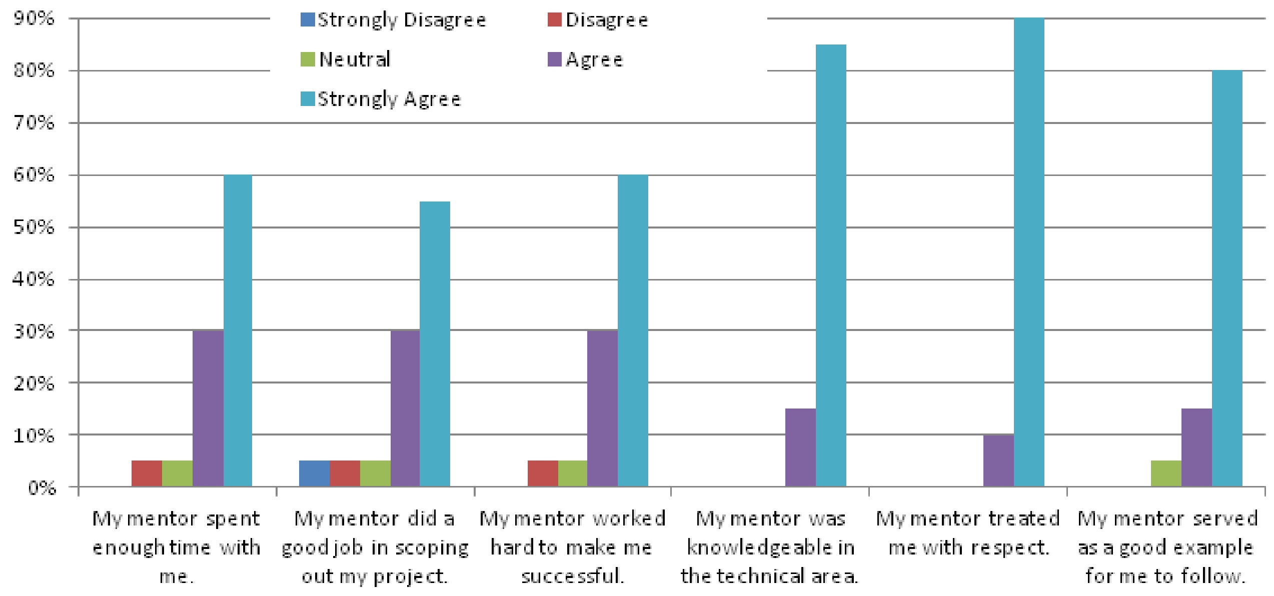
Summary of Improvements for Next Year: De-emphasize the use of Fuller Lodge and the Bradbury Science Museum in favor of using Research Park, spread out the furniture to reduce the noise level, discuss the noise level openly to provide guidelines, and add another meeting room.



Overall Result: The computing environment is reasonable, but needs upgrading.

Analysis: The above survey results indicate that the supercomputing facility Moonlight was acceptable, although perhaps not impressive, for meeting the students' needs. As in previous years, students used a mixture of the LANL-provided Sunray thin client for connections to Moonlight and their own laptops for other applications. There is wide disagreement on the effectiveness of the Sunrays; negative responses appear to be largely due to two factors, one being their speed and another being that the Sunrays' application software (e.g. LaTex and gnuplot) is out of date. These factors will be addressed before next year's workshop.

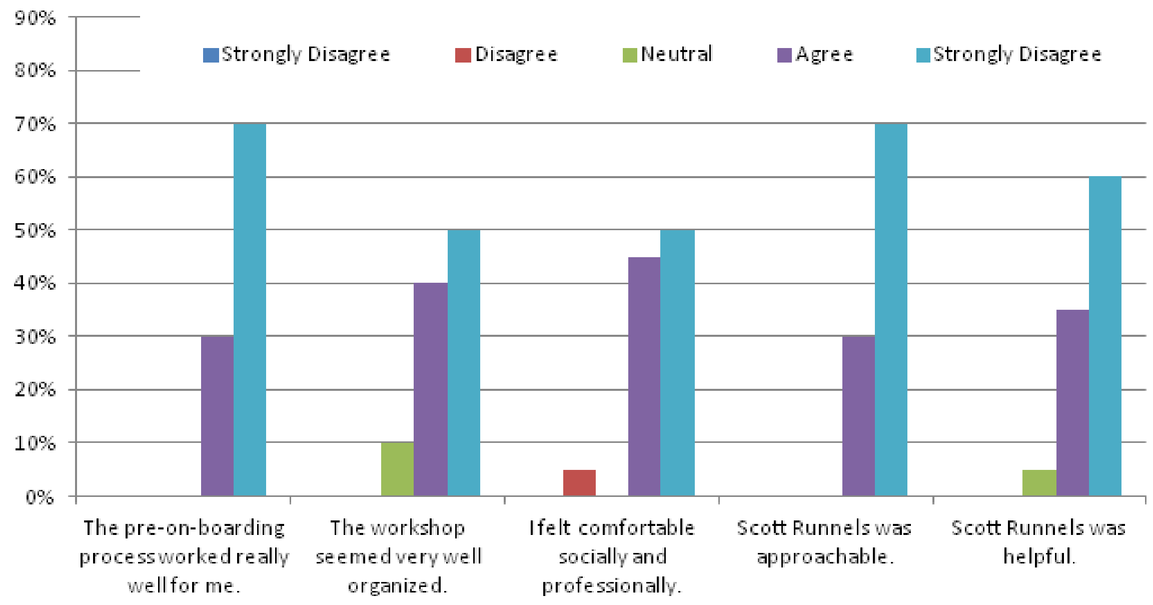
Summary of Improvements for Next Year: Improve the Sunray speed and upgrade the relevant software.



Overall Result: Mentors are excellent but we need to watch for weak spots.

Analysis: The above survey results indicate that the mentors, by and large, did extremely well with the students. This result is important because the mentor has a tremendous impact on the student's overall experience. There does appear to have been one mentor-student team that was weaker. However, the students that responded negatively also made the following comments, "I view this workshop as one of the best professional opportunities I have ever had," and "Overall really enjoyed the experience; it was educational, learned a lot. Regret not applying to LANL sooner." Therefore, while the above results do have some negativity associated with them, the individuals who provided the negative responses appear to have still viewed the workshop as a positive experience but are helping us improve by offering honest criticism.

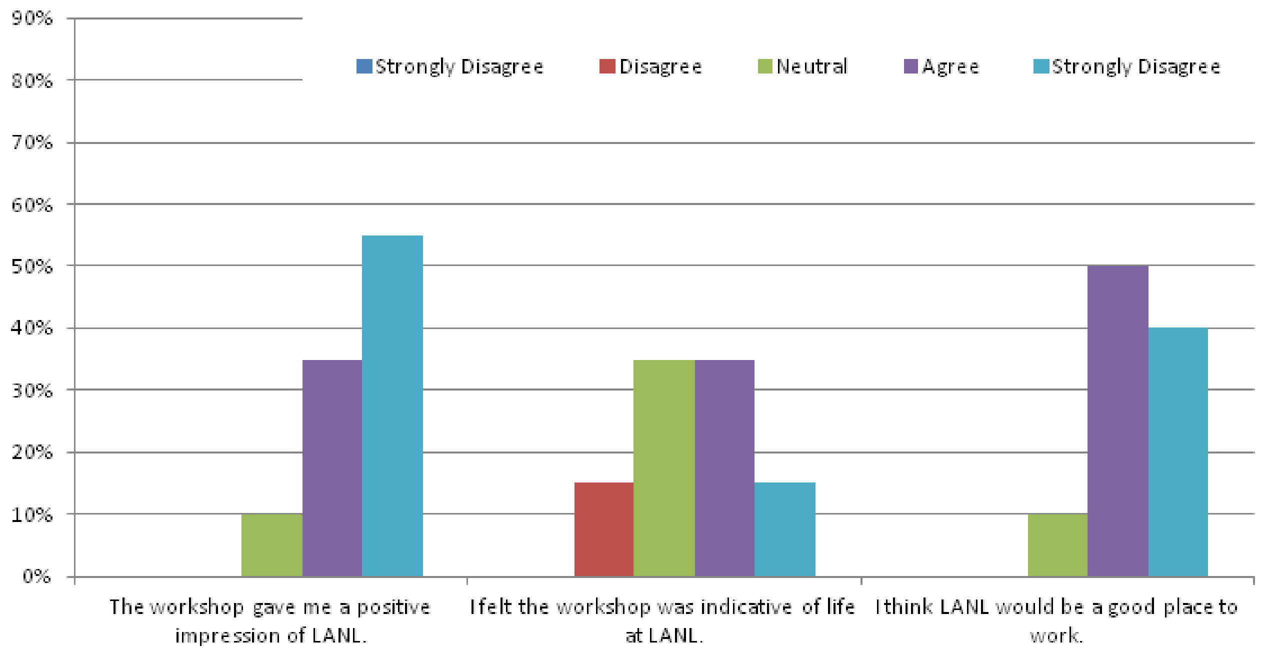
Summary of Improvements for Next Year: Establish a formal mid-summer checkup on the projects where students present on the status of their current project; this will improve focus for all teams, help prepare them for the Student Symposium, and also help identify weak spots.



Overall Result: The workshop organizer did well, but attention to social events should be renewed.

Analysis: The above survey results indicate that, overall, the organization of the workshop and the workshop organizer himself receive high marks. One negative response regarding the overall feel of the workshop, i.e., feeling comfortable socially and professionally, is cause for concern. However, the individual who gave the one negative response also wrote on their survey, "The workshop was a very valuable experience. I can think of no better way to have spent my summer." Therefore, the negative response is viewed as being from a student who had a positive experience but is trying to help us improve by offering honest criticism. There was also a comment about the lack of more social activities this year and these comments may be related. Also, it is speculated that the furniture was placed too close together in the new facility and that by spreading it out more next year, this negative comment as well as other negative comments about the conversation noise level will be remedied.

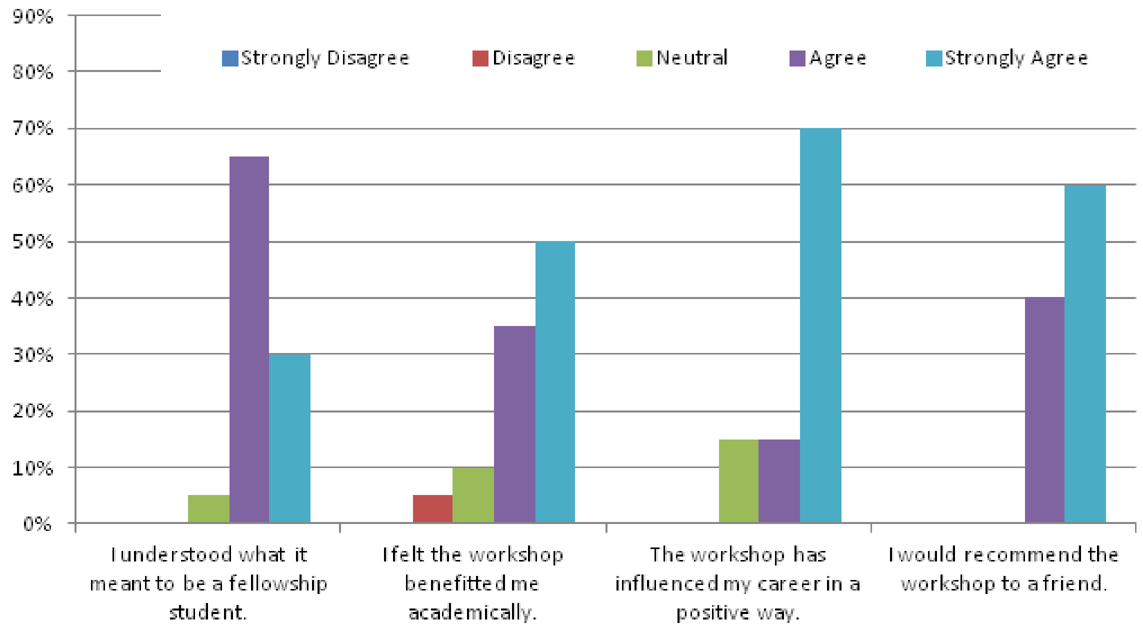
Summary of Improvements for Next Year: Spread out the furniture and renew attention to social activities to ensure they are being planned and executed.



Overall Result: The workshop leaves students with a positive impression of LANL.

Analysis: The above survey results indicate that, overall, the Summer Workshop left the students with a very good impression of LANL and with the feeling that they might like to work at LANL someday. However, that feeling is properly bridled by the students recognizing that the Summer Workshop is a special activity that is not necessarily reflective of the day-to-day work environment.

Summary of Improvements for Next Year: These results do not indicate a need for change beyond what has already been discussed above.



<u>**Overall Result:**</u> The workshop has a very positive impact on students' careers.

<u>**Analysis:**</u> The above survey results indicate that, overall, the workshop receives very high marks with respect to positively affecting the students' careers. The students seemed to understand the difference between being a Summer Workshop fellowship student and a regular student intern, that they learned a great deal, and that the workshop was successful in influencing their career in a positive way.

<u>**Summary of Improvements for Next Year:**</u> These results do not indicate a need for change that have not already been discussed, namely improving the academic experience by further improving the lecture series.

Introduction to the Technical Reports

As part of the Summer Workshop's emphasis on developing the participants' presentation and writing skills, they develop a comprehensive final report describing the project on which they worked during the summer. While these reports are integrated in this document, each chapter is intended to essentially be a stand-alone document. Nomenclature may not be consistent between the chapters; figures, equations, and concepts my be repeated.

Algorithms for Coalescing Shared Boundaries Between Parts

By Andrew Ho and Kerry Garrett, Davis Herring and Brian Jean (Mentors)

Abstract

To ensure correct connectivity in a conformal mesh all part-part boundaries must be identified. Currently it is entirely the user's responsibility to correctly identify all part boundaries. Because there may be numerous interfaces, an algorithm for finding 2D part boundary concurrencies has been developed which can automate this process. It is built on mathematical definitions for shared interfaces using a minimal set of numerical tolerances. We have demonstrated that the algorithm produces expected results in a variety of cases. From the outputs of this algorithm, it is possible to build an intersection graph that can be used to perform constructive solid geometry (CSG) operations on closed contours while preserving the topological information.

Motivation

Numerical methods for solving partial differential equations typically require discretization of the domain of interest into a mesh. Some types of discretization are trivial, for example dividing a 1D domain into a set of grid points (not necessarily uniformly spaced), generating a rectilinear 2D spatial grid, or using Fourier analysis to discretize the frequency domain. However, many interesting problems contain complex geometries and have multiple bodies. It is advantageous to use a conformal meshing algorithm because complex geometries can be represented accurately using fewer elements than other methods such as adaptive mesh refinement (AMR). There are several techniques for handling how multiple bodies interact:

No interaction The simplest solution is to do nothing. This is technically not physical because it allows physical bodies to interpenetrate. However, it may still be useful in situations where it is obvious two bodies will never interact in the simulation domain.

Joined meshes Joining two meshes means that shared mesh nodes are placed at the boundary. Each individual element can still be assigned to a single material/part, but the elements are no longer disjoint, and thus must move together.

Feature painting Feature painting is a slight variation on joined meshes. A key difference is that feature painting does not necessarily require mesh points at the interface. Instead, the interface is resolved using boundary reconstruction techniques. This allows relatively complex boundaries to be simplified when the exact shape is not crucial.

Surface contact resolution Surface contact resolution is by far the most expensive of the three solutions. This technique resolves surface contacts directly by examining the meshes and finding locations where they intersect. It is the most physical solution for preventing bodies from interpenetrate while still allowing them to separate when pulled apart, and may even be used to enforce "glued" bodies while maintaining disjoint meshes with multiple resolutions. However, despite being the most physical solution it requires a sufficiently resolved boundary mesh in order to accurately capture interfaces and expensive spatial checks and resolutions to handle actual interactions, thus for the same computational cost may not give as accurate results as some other solutions.

Multiple contact resolution techniques may be used in a single simulation. Figure 2 shows an example situation where different types of contact resolution are used to create a multibody simulation. For relatively simple cases, it is sufficient for a user to specify which faces could become coupled, and how the coupling should be resolved. However, this becomes unwieldy in large complex geometries. It is therefore advantageous to provide a semi-automated framework for identifying interface boundaries and querying the user for appropriate ways to resolve such boundaries.

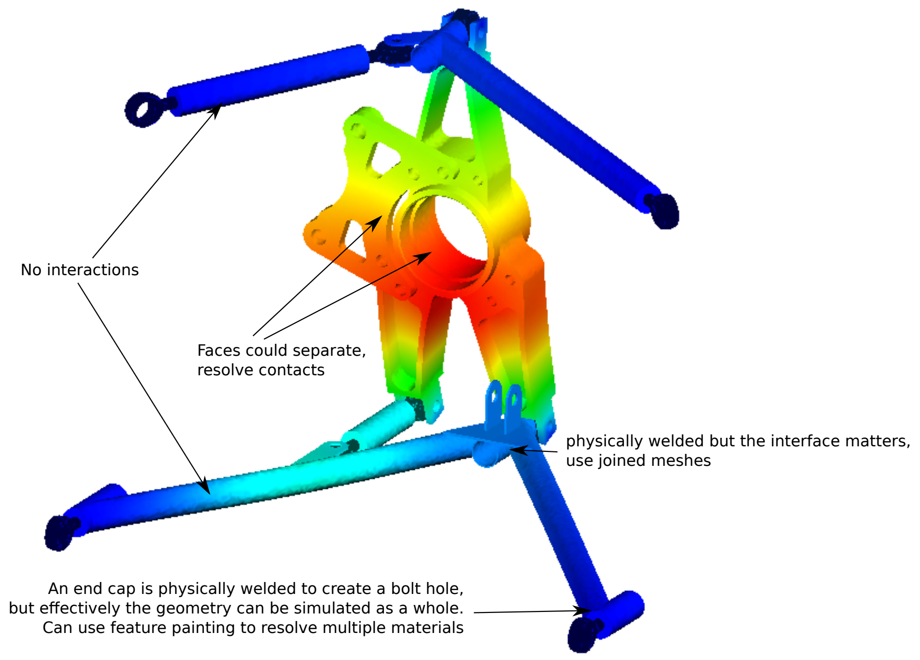

No interactions
Faces could separate, resolve contacts
physically welded but the interface matters, use joined meshes
An end cap is physically welded to create a bolt hole, but effectively the geometry can be simulated as a whole. Can use feature painting to resolve multiple materials

Figure 2: Multi-body simulation demonstrating different contact resolution techniques.

Project Scope

The framework is developed into LANL's Ingen software. This is a software package designed to create geometries, assign material properties, and generate meshes. There are various options for modeling 2D and 3D geometries. The primary area of use for this tool is for axisymmetric "semi-structured" meshes for use with Lagrange Hydro codes. The term "semi-structured" is used to indicate that the mesh generation is done using a structured meshing algorithm, but additional processing on the mesh may result in unstructured meshes, as shown in Figure 3. This process is detailed in [32].

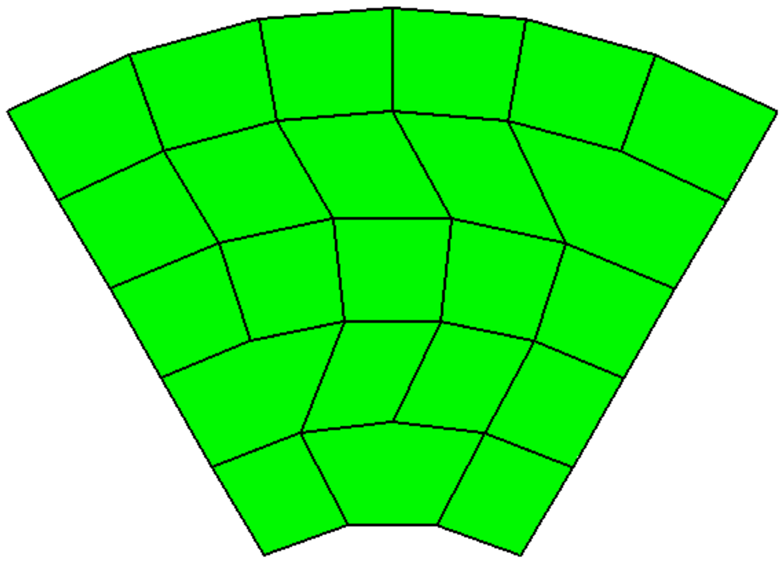


Figure 3: Semi-structured mesh. The original structured mesh has had dendrites applied.

Additionally, multiple blocks may be meshed to be internally structured, but the complete model is not structured (or even semi-structured). However, the edges between parts still share common vertices. This is demonstrated in Figure 4.

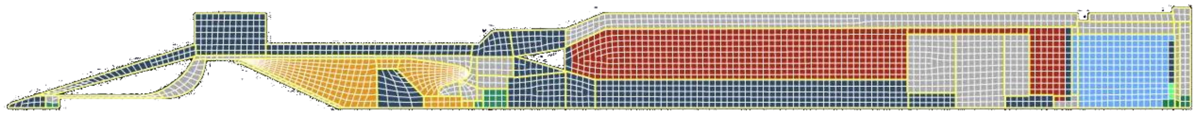


Figure 4: Multiple connected mesh blocks.

There are 2 types of potential geometry input:

- 3D constructive solid geometry (CSG)
- 2D piecewise linear contours, which are assumed to be revolved around an axis.

In this project we used both the geometric and topological information to construct a solution which is both general and composable. The solution is designed to be robust against numerical noise.

Constructive Solid Geometry (CSG) approach

Constructive solid geometry (CSG) is a technique for defining complex shapes in terms of simple primitive volumes using various set operations. This means that fairly complex shapes can be designed from simple components and all intermediate and final results are guaranteed to be closed surfaces. CSG region definitions take the form of a directed acyclic graph (DAG); regions may be referenced by any number of operators but not by themselves. An example is provided in Figure 5.

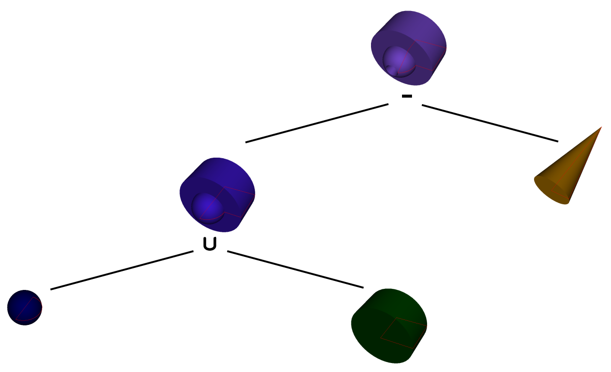


Figure 5: example of CSG representation and producing DAG

Because CSG is built using set operators a resulting region can only ever contain points found in any one of the primitives related in the tree. That means surface representations must fundamentally be built from sub surfaces of the underlying primitives. Additionally, sub surfaces can only ever be created where two primitive surfaces intersect or coincide.

An axisymmetric surface sliced into a 2D plane results in contour(s). If there is only one resulting contour the shape is said to be simply connected in an axisymmetric sense. However, CSG regions (or their 2D slices) need not be simply connected. Figure 6 shows it is entirely possible for simply connected shapes to be formed from intermediate CSG regions which are not simply connected, and that it is also possible to create non-simply connected shapes using only simply connected primitives. Note that there additionally are regions which are not simply connected, but which have a simply connected axisymmetric slice. An example case is a torus.

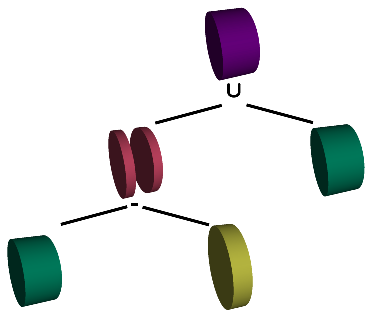


Figure 6: Transforming between simply connected and non-simply connected regions via CSG operations.

Thus, there are only two situations where a face could become shared:

1. Two CSG objects share a child primitive
2. Two CSG objects share child primitives which share a common surface

It is important to note that, due to axisymmetry, two primitives can share a common side at the axis of revolution as well.

Topological Approach

Boundary representation (B-rep) models use both geometry and topology. A geometric representation of a solid is composed of surfaces (2D), curves (1D) and points (0D), all having a specific location in space. The analogous topological components of the solid are the faces, edges, and vertices. Other topological entities exist such as a loop (alternating sequence of vertices and edges), body (independent solid), and genus (hole), but for brevity are not discussed here. A simplified topology–geometry connection between components is given as follows: a face is a bounded, non-self-intersecting part of a surface, an edge is a non-self-intersecting segment of a curve bound by two vertices, and a vertex represents a unique point in space. With topology, higher-order objects can share lower-order objects, a point that is illustrated by topological concurrencies. In the B-rep model, geometry is added so to define the object's spatial position. For instance, two separated spheres, one with smaller radius than the other, are topologically equivalent to the two spheres with one being nested in the other since their location in space is not topologically relevant. In the full B-rep model,with coordinates added, the nested spheres would result in a solid with two faces, whereas the disjoint spheres would represented as two distanced solids. The topology of the B-rep model includes information about how the faces, edges, and vertices are interconnected. Thereafter, geometric information for each face, edge, and vertex is developed via equations and coordinates. A major advantage of B-rep is that through knowing how the faces, edges, and vertices of an object are connected, one then has a description of the bounding surface, an important feature which CSG does not provide.

Conversion of CSG to Topologically-Informed Model

CSG is a purely geometric representation of a solid since its operations are chosen for compatibility with point queries. This representation does not provide topological information about the solids, hence, conversion from CSG to a boundary representation (B-rep) is non-trivial. One method of converting CSG to B-rep is a *membership classification algorithm* [7]. The B-reps of the primitives are developed recursively such that they are passed up the tree to the root. This requires that, at each node, the B-reps be classified and combined according to the given operation at each node. In particular, computing an operation on two solids in B-rep requires finding the intersections between the boundaries of each solid. Henceforth, the algorithm requires classification of the faces of each B-rep primitive to determine whether one is inside, outside, or on (sharing a boundary with) the other. The algorithm generally goes as follows:

- Define the faces for each primitive, *A* and *B*
- Find all edges of each operand *A* and *B*
- Classify all edges according to each face:
  - inside, outside, or on *A*
  - inside, outside, or on *B*
- Select edges based on the *boolean operation*:
  - **Union**: remove edges *in A* & *in B*
  - **Intersection**: remove edges *out of A* & *out of B*
  - **Difference**: remove edges *on A* & *in B* and *out of A* & *in B*

The *intersection graph* algorithm described in this section an alternative CSG to B-rep conversion algorithm. The algorithm works in 2D, such that contours are first obtained for each of the volumetric primitives, as shown in figure 7. To have topological information survive through CSG operations requires storing each of the contours in a separate data type. After the contours for each primitive region have been defined, each contour is assigned a pointer, which stores parts of each contour in a shared location from which multiple sides can be generated. Technically, the sides are stored as a reference and a parameter range corresponding to the underlying contour. Sides can be joined to make structures from multiple contours, called a chain (or a perimeter if it is closed). The current implementation of intersection graph code requires that the CSG operations be performed on a closed region, thus the user can build perimeters from the region(s) for which they want to perform the CSG operations.

After the user has defined their region(s), perimeters are constructed for each primitive in the DAG. The intersection graph contains information about the operands, edges, and vertices. The algorithm begins in a similar fashion as outlined above: identify an intersection (node); find all edges of operand *A* and *B* about this node and color the edges according to operand; classify each edge according whether it is inside and/or outside of *A* or *B*. The algorithm considers three main cases (eight different states total) to select for each edge at a node:

Case 1. the trivial case that no edge exists,

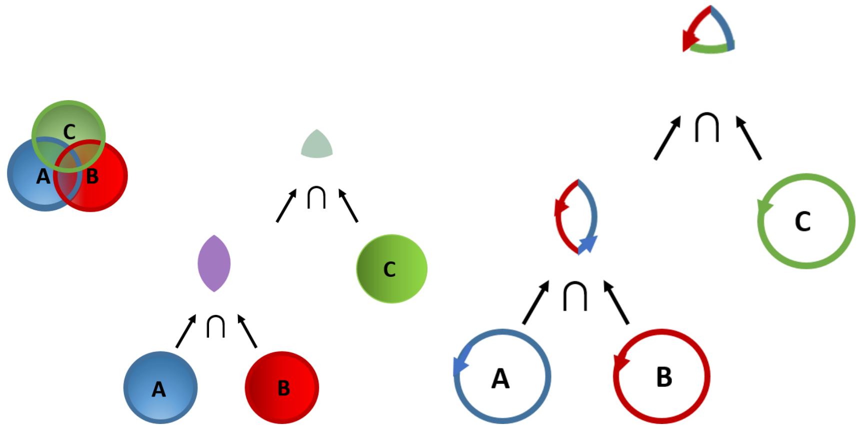
Case 2. the standard case in which the contours do not overlap,

- ingoing (left), outgoing (left), ingoing (right), outgoing (right)

Case 3. the case in which there is overlap

- ingoing (concurrent), outgoing (concurrent), opposite (wrong-way) concurrency

It is necessary to identify the direction of the edge in order to determine the direction of *sweep*. The algorithm effectively sweeps along each node to decide which edge to keep or remove based on the boolean operation. The algorithm uses tangents to pass through each edge. The direction of sweep is either clockwise (cw) or counter-clockwise (ccw). For example, in this implementation, an outgoing right edge sweeps ccw; whereas, an ingoing edge from the left sweeps cw. The algorithm works recursively so that operations between primitives on subtrees are handled first and combined up the tree as the algorithm journeys to the root. Once the root is reached, the end result is a composite perimeter containing information about each primitive's contribution to the boundary. The desired object is created and its topology is intact.



(a) CSG approach. (b) Topological approach.

Figure 7: Example comparing (a) CSG model and (b) Topological model for constructing the intersection of three spheres.

In the following, we examine Case 2. Referring to figure 7, we wish to obtain the intersection of three spheres using the topological model. The intersection graph algorithm is recursive such that we begin with the operands A and B, as shown in figure 8. In this illustration, the operands A and B are distinguished via color; in code, the identity of the contour object is used. Referring to figure 8, the top vertex (shown in yellow) has an outgoing blue edge corresponding to operand A and a red left outgoing edge corresponding to operand B. The desired operation is $A \cap B$, hence, beginning with the tangent on the outgoing edge of operand A the algorithm sweeps counterclockwise through the upper right quadrant, $\cup^c$, in which neither A or B exist; it then continues to $B - A$. The edges not partipating in the operation $A \cap B$ (*in A* & *in B*) are removed. Once the sweep reaches the lower left quadrant, the condition is satisfied and the ingoing blue and outgoing red edges are saved. To complete the rotation the upper left quadrant, $A - B$, is examined and the remaining edges are removed. The algorithm sweeps through until the cycle has completed. Similarly, the bottom vertex (shown in green) is evaluated. The resulting object is a wedge of the two intersecting contours. The left side of the boundary comes from operand B and the right side from operand A. It is then understood that the two boundaries intersect where the two colored contours touch, thus the topologically relevant information is stored. The algorithm moves recursively up the tree to the root. The next step is to combine the product of the previous operation with that determined by the next node and its children. Similar to the operation on A and B, the operation between $(A \cap B) \cap C$ is now performed, as shown in figure 9.

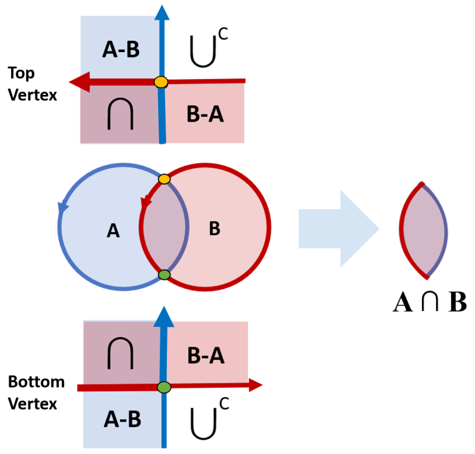


Figure 8: Selecting edges at each intersection of operands A and B.

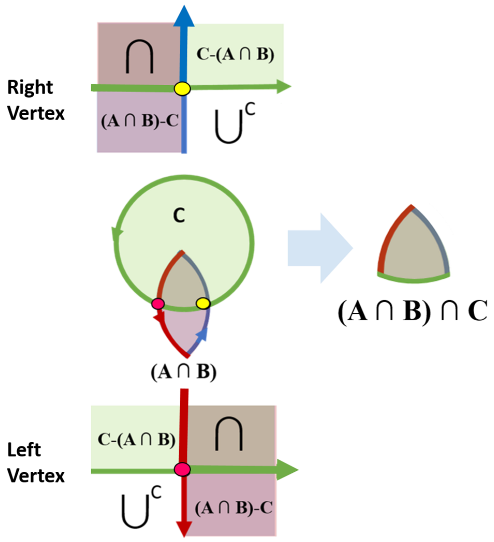


Figure 9: Moving up the tree; including operand C in intersection.

The intersection graph algorithm relies on communication between various classes and functions. In summary, the user calls the CSG function when they choose an operation, Union (|), Intersection (&), or Difference (-), between two perimeters. The CSG function uses an *intersection finder* class, which creates a list of all intersections between the two perimeters; this class also calls an *intersection tracking* class to mark each time an intersection is visited, as well as a *filter function* which removes edges with each sweep. The *intersection tracking* class uses the class *Edge* which assigns a color to each operand, direction of edge (ingoing/outgoing), to determine the direction of the tangent used in each edge-reducing sweep.

Benefits of the Topological Model

As mentioned previously, a major advantage of using a topological model instead of CSG is to obtain a complete description of the bounding surface of an object. In the example shown in figures 7−9, the resulting solid (intersection of three spheres) contained information about which primitive contributed to which part of its boundary. Unlike CSG, the intersections between the boundaries are known. An additional benefit of the topological model is the ease with which topological concurrencies are handled. A simple example is shown in figure 10. The subtrees describe the difference between a sphere and a cylinder. The boundary that the cylinder (C) contributes to the intermediate object (half circle) does not require any special handling at the next node for the intersection, $(A-C) \cap (B-C)$. The 'green' boundary from C is common between the intermediates $(A-C)$ and $(B-C)$ so that exactly one shared boundary exists. The boundary in the final object is simply treated as a contribution from the cylinder, C, just as it was discovered in the subtree.

The intersection algorithm described above handles topological concurrencies, in which the algorithm encounters < 4 edges. In this case, the algorithm identifies the three edges and sweeps ccw from the outgoing blue curve $(A-C)$, here labeled as D, includes the $\cap$ in the result and similarly excludes the edges that do not contribute to the operation. In this example, the edges in the region of $E-D$, where E is the abbreviated form of $(B-C)$, are discarded.

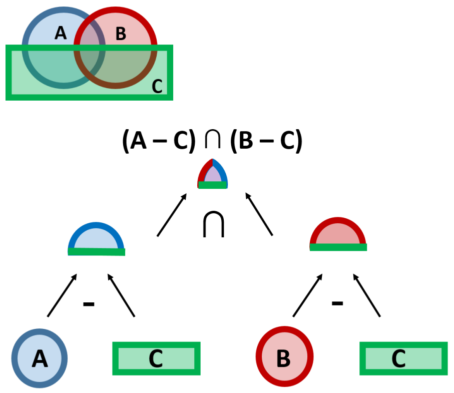


Figure 10: Example of Topological Concurrency

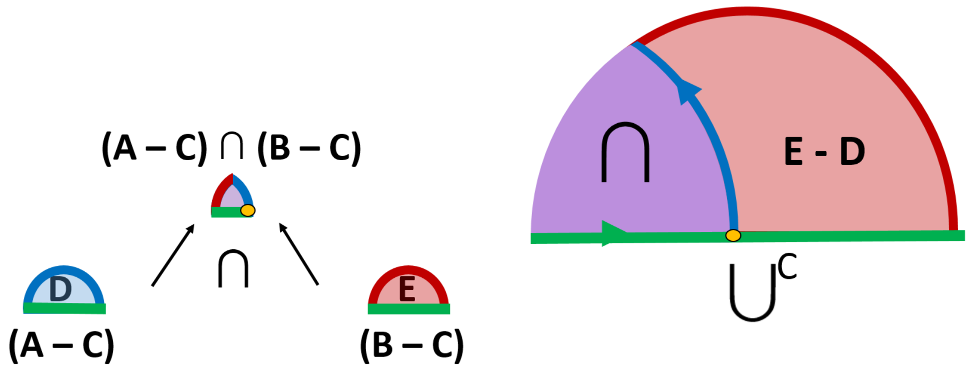


Figure 11: Intersection algorithm handles the case of < 4 edges

Despite the clear advantages of the topological model, it is not always possible to neglect geometry. For instance, in case two boundaries are within some small tolerance of each other, a geometric definition of the boundaries may be necessary. In such cases, it is necessary to distinguish between a set tolerance that defines geometric concurrency and numerical noise. It is possible to then convert geometric concurrencies to topological concurrencies using the intersection graph algorithm. The geometric approach and numerical tolerances are discussed in the following section.

Geometric Approach

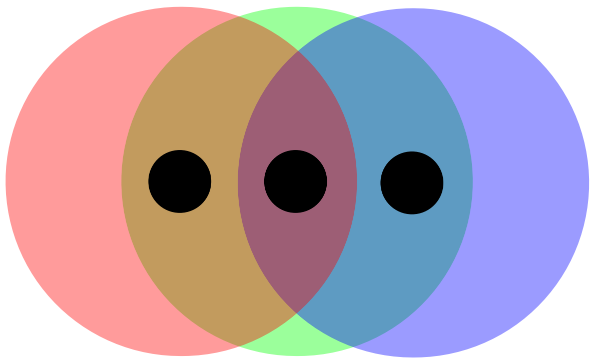
Some topological information must be derived from geometric analysis. The relevant information which can only be deduced by analyzing the geometry is:

- Location of interface coincidences and intersections.
- Whether a region is inside/outside another region.
- Surface tangents.

It is also at the geometric level that numerical tolerances must be considered. A proper choice of numerical tolerances is crucial for ensuring the solution is self-consistent and mathematically rigorous.

Choosing Numerical Tolerances

Numerical tolerances are an unfortunate necessity for finite precision computation. The use of any numerical tolerance will always create non-transitive equality. Figure 12 shows two possible choices for numerical tolerances and associated non-transitive equality examples.


(a) Distance-based tolerance

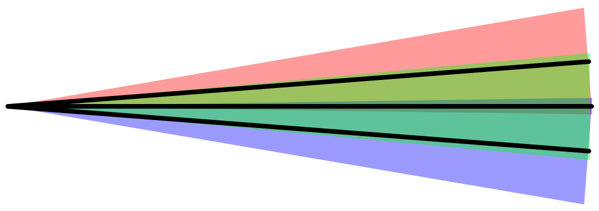

(b) Angle-based tolerance

Figure 12: Non-transitive equality for distance and angle-based tolerances.

Additionally, combining multiple tolerances can lead to compounded problems where the various tolerances do not agree on whether two items are within tolerance or not. Consider Figure 13, where angle tolerances would rule the two segments as parallel, but a distance tolerance would rule the segments as sharing an intersection or a different coincidence. Because of these two negative properties, it is desirable to pick as few numerical tolerances with actual physical significance to the input data provided by the user.

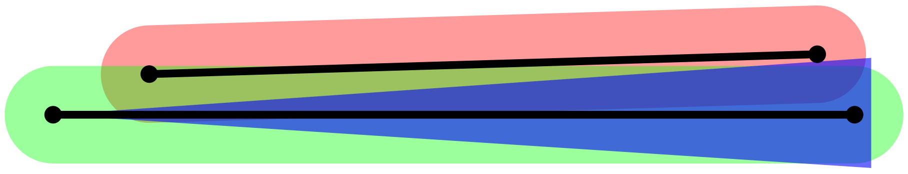


Figure 13: Disagreement between multiple tolerances.

By choosing tolerances based on the input data it is meaningful to demand a consistent scale separation between numerical noise and physically significant data. By having a scale separation the problem of having to deal with non-transitive equality disappears because it is simply assumed that a user will not provide physically significant dimensions within a few orders of magnitude near the numerical tolerance. This is demonstrated in Figure 14. D is the largest model dimension and d is the smallest model dimension. Based off of D, numerical noise is computed to be ε. The blue region highlights all possible tolerances which respect the scale separation. Any single tolerance value δ may be chosen in this region which will not have issues with numerical noise or non transitive equality. As the scale disparity increases, the blue region may shrink or grow. If the blue region disappears entirely, the user has violated the scale separation requirements and no tolerance will be acceptable.

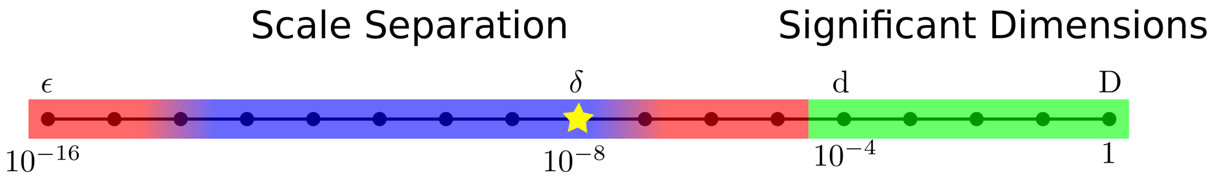


Figure 14: Scale separation requirements and viable tolerances.

Line Segment Coincidences/Intersections

The existing line segment coincidence finder algorithm is based on the method developed by Goldman (1990) [24]. A rough outline of this method is provided below.

A line can be defined as a starting location p and a vector $\vec{v}$ indicating direction. Two lines intersect iff there is a unique solution to the equation:

$$\vec{p}_1 + t\vec{v}_1 = \vec{p}_2 + u\vec{v}_2 \tag{1}$$

In Goldman's method these separated are into two equations to solve for the parameter t and u:

$$t = \frac{(\vec{v}_1 \times \vec{v}_2) \cdot ((\vec{p}_2 - \vec{p}_1) \times \vec{v}_2)}{\|\vec{v}_1 \times \vec{v}_2\|^2} \tag{2}$$

$$u = \frac{(\vec{v}_1 \times \vec{v}_2) \cdot ((\vec{p}_2 - \vec{p}_1) \times \vec{v}_1)}{\|\vec{v}_1 \times \vec{v}_2\|^2} \tag{3}$$

The last step is to recognize that a line segment is simply a bounded range on the line, i.e. $t_l \leq t \leq t_u$. These methods recognize that the existence of a unique solution pair t, u is predicated on the fact that the two lines are not parallel or near parallel. If the only possible case for ruling the lines as being coincident is if the two lines are exactly parallel, then these methods can be used successfully. However, the problem becomes extremely ill-conditioned in the case where lines become nearly parallel.

Traditionally a tolerance on how parallel the two lines are is used. However, parts of the algorithm rely on a distance-based tolerance in order to check if the line intersection point is on the line segments or not.

Rather than deriving an exact solution and then modifying it to account for numerical tolerances, our solution takes into account the numerical tolerances to define what it means for two segments to coincide or have an intersection. We define a *same point tolerance* (SPT) as the minimum distance between two physically distinct points. For a line segment $\overline{AB}$, we can express the SPT region $\Omega(\overline{AB})$ of size δ as:

$$\Omega(\overline{AB}) = \{\vec{x} \in \mathbb{R}^n : min\|\overline{AB} - \vec{x}\| < \delta\} \tag{4}$$

Two line segments $\overline{AB}$ and $\overline{CD}$ are then defined to share a coincidence iff it is possible to find two physically distinct endpoints each in the SPT region of the other segment. Two segments can then be defined to have an intersection iff they share no coincidences and:

$$\exists F \in \overline{AB} : F \in \Omega(\overline{CD}) \tag{5}$$

Examples of segment-segment coincidences and intersections are shown in Figures 15 and 16 respectively.

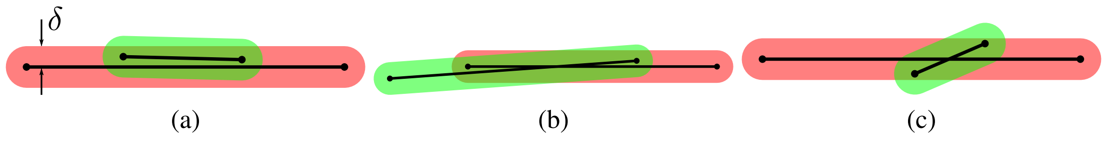


Figure 15: Example of coincident segments

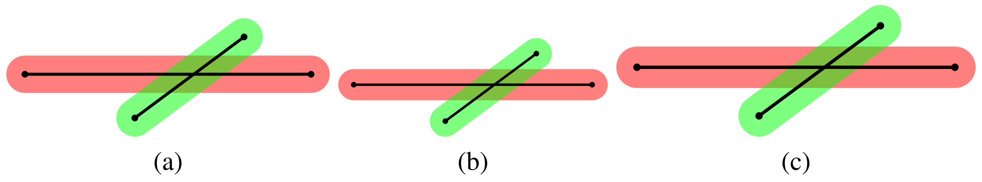


Figure 16: Example intersecting segments

Notice that with this definition we end up with unfortunate cases such as 15(c), where segments with a well behaved line intersection solution are defined to be coincident. However, this is deemed to be acceptable because such cases violated the initial assumption that there must be a scale separation between numerical noise, the chosen tolerance, and physically significant dimensions. This definition's single tolerance avoids the self-consistency issues previously discussed. An implementation of this algorithm can be constructed using vector projections as shown in Algorithm 1.

Algorithm 1 Segment coincidence/intersection finder

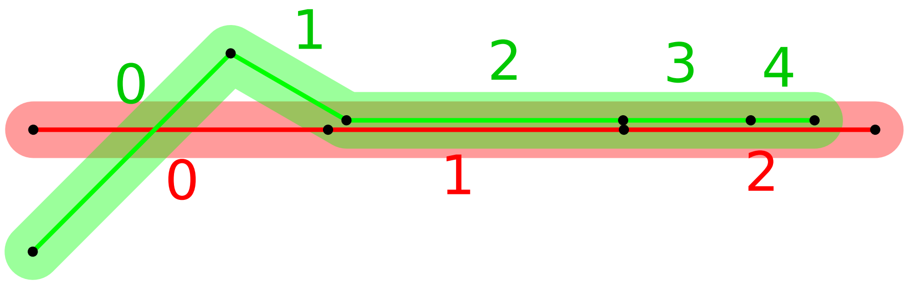
```
**Require:** $\vec{p}_{11}, \vec{p}_{12}, \vec{p}_{21}, \vec{p}_{22}$
1: $\vec{v}_1 = \vec{p}_{12} - \vec{p}_{11}$
2: $\vec{v}_2 = \vec{p}_{22} - \vec{p}_{21}$
3: $\vec{v}_{11} = \vec{p}_{11} - \vec{p}_{21}$
4: $\vec{v}_{12} = \vec{p}_{12} - \vec{p}_{21}$
5: $\vec{v}_{21} = -\vec{v}_{11}$
6: $\vec{v}_{22} = \vec{p}_{22} - \vec{p}_{11}$
7: $r = \|\vec{v}_1\|^2$
8: $s = \|\vec{v}_2\|^2$
9: $uv_1 = \frac{\vec{v}_{11} \times \vec{v}_2 \cdot \hat{z}}{s}$
10: $uv_2 = \frac{\vec{v}_{12} \times \vec{v}_1 \cdot \hat{z}}{s}$
11: $tv_1 = \frac{\vec{v}_{21} \times \vec{v}_2 \cdot \hat{z}}{r}$
12: $tv_2 = \frac{\vec{v}_{22} \times \vec{v}_1 \cdot \hat{z}}{r}$
13: $tol1 = \|\vec{v}_1\|^{-1}$
14: $tol2 = \|\vec{v}_2\|^{-1}$
15: $d = [\|\vec{v}_{21}\|, \|\vec{v}_{22}\|, \|\vec{v}_{12}\|, \|\vec{p}_{22} - \vec{p}_{11}\|]$
16: num_near = count(abs([$tv1, tv2$]) $< tol1$) + count(abs([$uv1, uv2$]) $< tol2$) − count($d <$ SPT)
17: **if** num_near $\geq 2$ **then**
18:     $u_1 = \frac{\vec{v}_{11} \cdot \vec{v}_2}{s}$
19:     $u_2 = \frac{\vec{v}_{12} \cdot \vec{v}_2}{s}$
20:     $t_1 = \frac{\vec{v}_{21} \cdot \vec{v}_1}{r}$
21:     $t_2 = \frac{\vec{v}_{22} \cdot \vec{v}_1}{r}$
22:     **if** $-tol2 < u_1 < 1 + tol2$ or $-tol1 < t_1 < 1 + tol1$ or $-tol1 < t_1 < 1 + tol1$ or $-tol1 < t_2 < 1 + tol1$ **then**
23:         $t_1, t_2 = \min(t_1, t_2), \max(t_1, t_2)$
24:         **if** $abs(u2 - u1) \geq tol2$ and $(t_2 - t_1) \geq tol1$ **then**
25:             **return** $(t_1, t_2), (u_1, u_2)$
26:         **end if**
27:     **end if**
28: **end if**
29: **if** $uv_1 \neq uv_2$ and $tv_1 \neq tv_2$ **then**
30:     $t = \frac{uv_1}{uv_1 - uv_2}$
31:     $u = \frac{tv_1}{tv_1 - tv_2}$
32:     **if** $-tol1 < t < 1 + tol1$ and $-tol2 < u < 1 + tol2$ **then**
33:         **return** $t, u$
34:     **end if**
35: **end if**
36: **if** Any endpoints near each other **then**
37:     **return** endpoint parameters
38: **end if**
39: **return** None
```

Contour Coincidences and Intersections

Even though contours are defined as chains of line segments, it is unhelpful to return as output that two arcs share tens or hundreds of segment coincidences/intersections. Instead, the neighboring results from Algorithm 1 should be coalesced if they are near. This is done by recognizing that results can only be coalesced if the involved segments are adjacent. By making finding adjacent results quick, it is possible to combine adjacent results without having to check every possible combination. This can be done be creating a matrix whose row indices coorespond to one contour's segments, and the columns correspond to the other contour's segments. At every combination of segment pairs there can only be one unique result:

- a coincidence
- an intersection
- nothing

Neighboring events are logically one element away in any direction and can be quickly found. This is demonstrated in figure 17. In general results will be sparse, thus a coordinate sparse matrix storage format is used. Neighboring events should only be merged if by geometric inspection the events are sufficiently close. For example, the intersection event in $(0,0)$ cannot be merged with any other event despite having two neighboring events. Algorithm 2 gives the general outline for merging coincidences. Figure 18 demonstrates graphically how the results of 17 would be merged.

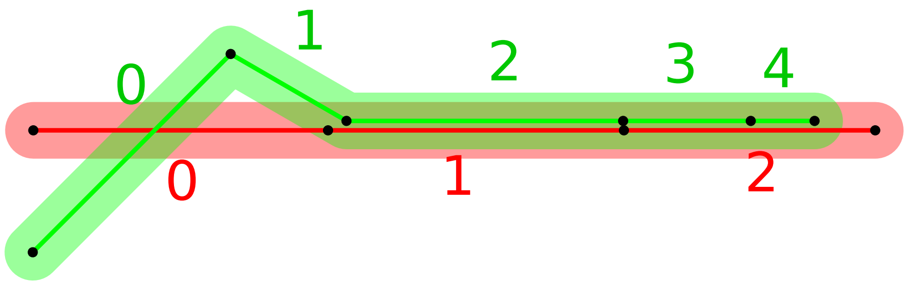


$$\begin{array}{c|ccccc}
 & 0 & 1 & 2 & 3 & 4 \\
\hline
0 & I & I & I & & \\
1 & & I & C_1 & I & \\
2 & & & I & C_2 & C_3
\end{array}$$

Figure 17: Uniquely identifying coincidence/intersection events.

Algorithm 2 Contour coincidence merger

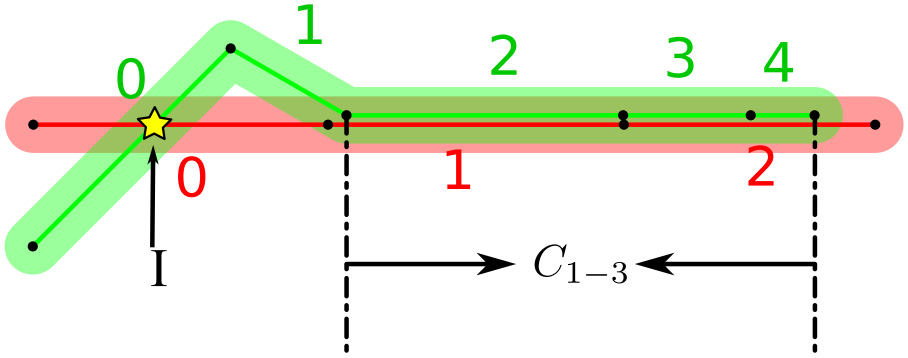
```
Require: cntr_A, cntr_B, seg_intersections, seg_coincidences
 1: coincidences = {}
 2: intersections = {}
 3: for all $I_{i,j}$ in seg_intersections do
 4:     intersections[$(i,j)$] = $I_{i,j}$
 5:     remove all close neighbors of $(i,j)$ in intersections
 6: end for
 7: for all $C_{i,j}$ in seg_coincidences do
 8:     remove all close neighbors of $(i,j)$ in intersections
 9:     for all neighbors $C_n$ of $(i,j)$ in coincidences do
10:         if $C_n$ is near $C_{i,j}$ and the coincidences are the same direction then
11:             $C_{i,j}$ = join($C_n$, $C_{i,j}$)
12:             Change all references of $C_n$ in coincidences to $C_{i,j}$
13:         end if
14:     end for
15:     coincidences[$(i,j)$] = $C_{i,j}$
16: end for
17: collect all unique results in coincidences and intersections
18: return coincidences, intersections
```



	0	1	2	3	4
0	I	I	I		
1		I	C_1	I	
2			I	C_2	C_3

$\Rightarrow$

	0	1	2	3	4
0	I	$\varnothing$	$\varnothing$		
1		$\varnothing$	C_{1-2}	$\varnothing$	
2			$\varnothing$	C_{1-2}	C_3



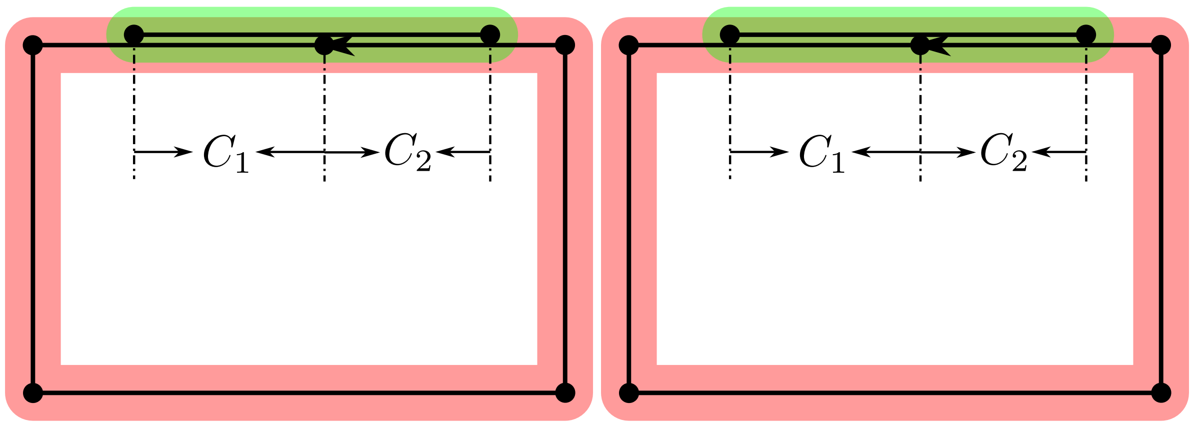
	0	1	2	3	4
0	I				
1			C_{1-2}		
2				C_{1-2}	C_3

$\Rightarrow$

	1	2	3	4	5
1	I				
2			C_{1-3}		
3				C_{1-3}	C_{1-3}

Figure 18: Merging process.

Periodic contours are contours where the first and last point are the same. Without modifying the original contour, it is only possible to construct sides which do not cross the periodic point. Therefore, a coincidence is only considered to be a neighboring event strictly based off of segment indexing. Intersections are usually concidered to be split points. There can only be one split point at any point in physical space, therefore merging of intersections should ignore periodic boundaries. This is demonstrated in Figure 19, where the two coincidences do not get merged, however intersections do.



(a) Coincidences are not merged. (b) Intersections are merged.

Figure 19: Coincidences and intersections across a periodic point.

Accounting for sides

A side is a view of a portion of a contour. Multiple sides may be allocated for a contour, and sides may overlap with other sides on the same contour. An example of two sides with their respective contours is shown in Figure 20. In addition to having to discard/reduce certain results, it is possible for contour coincidence results to be converted to side intersection results.

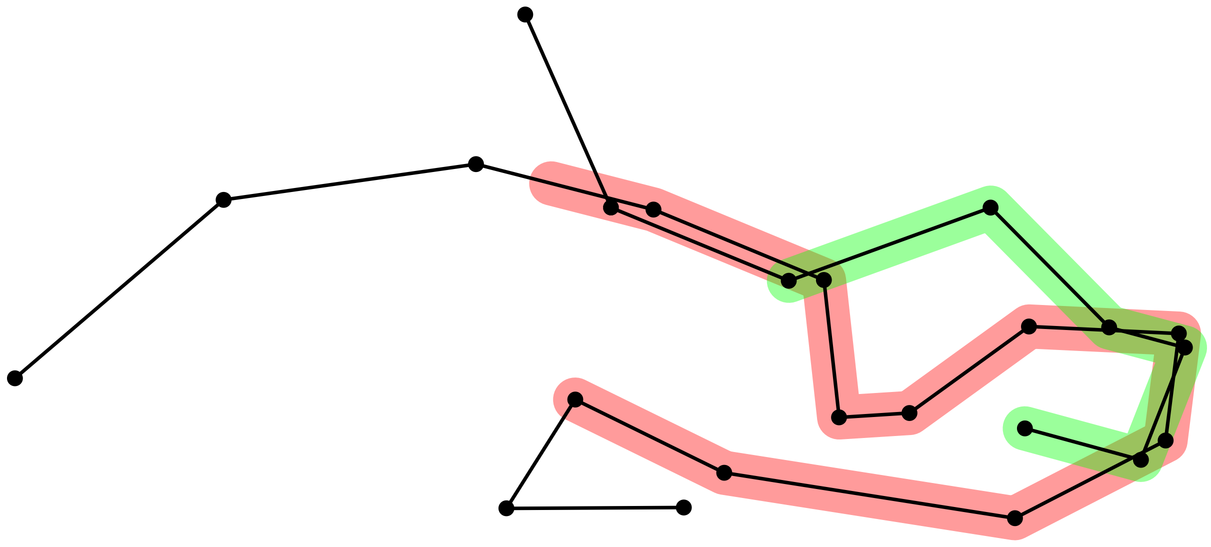


Figure 20: Coincidences and intersections of sides.

Taking into account sides may occur before or after. The trade-offs are that filtering for side results before may present fewer results to the contour merger algorithm, but adds to the complexity of the coincidence finder. For implementation details we decided to implement the post filtering process. The basic idea is to parameterize position along a contour by a single normalized scalar quantity. The position at the start of the contour is at $t = 0$, and the position at the end of the contour is at $t = 1$. All sides and interactions can then be phrased as the portion of the contour they involve. A side spans the range $[t_1, t_2]$, an intersection involves the point (t, u) (where t is the parameter on one contour and u is the parameter on the other contour), and a coincidence spans two ranges on two contours (ts, us) (where ts is a side on one contour and us is a side on the other contour). Filtering is simply a process of finding the overlap between the side span and any interaction event, and transposing coincidence overlaps into intersections if their span shrinks to near 0. This is detailed in Algorithm 3.

Algorithm 3 Side coincidence and intersection finder

```
Require: cntr$_A$, cntr$_B$, span$_A$, span$_B$
 1: coincidences, intersections = cntr_interactions(cntr$_A$, cntr$_B$)
 2: for all intersections $i$ do
 3:    if $i.t$ is not in span$_A$ or $i.u$ is not in span$_B$ then
 4:       remove $i$
 5:    end if
 6: end for
 7: $r_{AB} = \frac{\text{len}(\text{span}_A)}{\text{len}(\text{span}_B}$
 8: $r_{BA} = \frac{\text{len}(\text{span}_B)}{\text{len}(\text{span}_A}$
 9: for all coincidences $c$ do
10:    if $c.ts$ overlaps span$_A$ then
11:       $s_A = [\max(0, \min(\text{span}_A) - \min(c.ts)), -\max(0, \max(\text{span}_A) - \max(c.ts))]$
12:       $c.ts = c.ts + s_A$
13:       $us = \text{sort}(c.us) + s_A * r_{AB}$
14:       if $us$ overlaps span$_B$ then
15:          $s_B = [\max(0, \min(\text{span}_B) - \min(us)), -\max(0, \max(\text{span}_B) - \max(us))]$
16:          $us = us + s_B$
17:          $c.ts = c.ts + s_B * r_{BA}$
18:          if len$(c.ts) < tol$ then
19:             intersections.append((c.ts[0], us[0]))
20:             remove $c$
21:          else if reversed$(c.us)$ then
22:             $c.us = \text{reverse}(us)$
23:          else
24:             $c.us = us$
25:          end if
26:          continue
27:       end if
28:    end if
29:    remove $c$
30: end for coincidences, intersections
```

Results

As a realistic test of the algorithm, a high-explosive anti-tank (HEAT) shell model shown in Figure 21 was created using Ingen's solid modeling toolkit. Using existing tools, the part boundaries were then extracted as individual contours (Figure 22). Finally, a brute force comparison between every part boundary pair was performed to find all shared part boundaries (Figure 23). The algorithm identified 70 coincidences and 11 intersections [1]. The entire operation took a few seconds to complete.

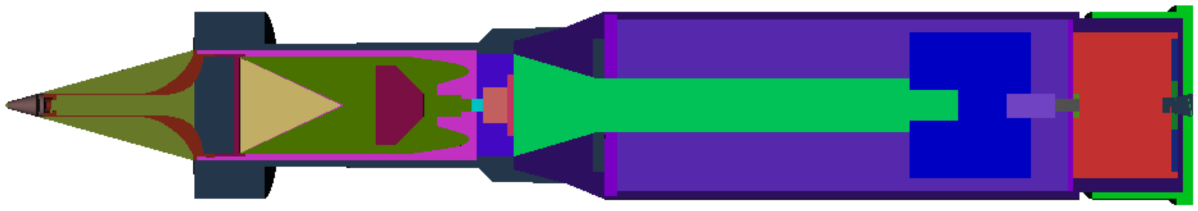


Figure 21: HEAT shell regions.

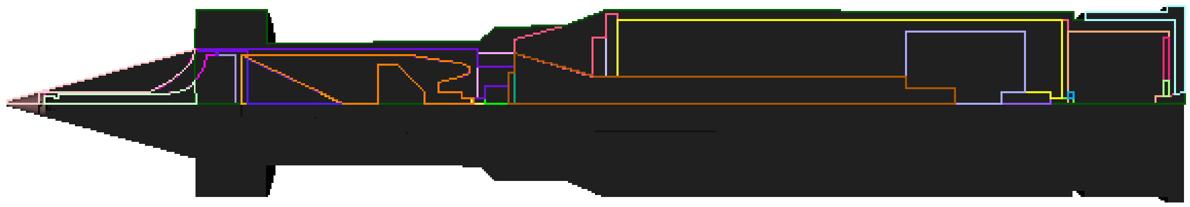


Figure 22: HEAT shell part boundaries.

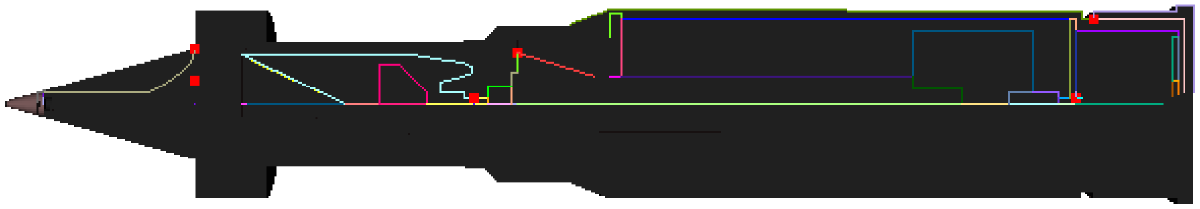


Figure 23: HEAT shell shared boundaries (coincidences and intersections).

Conclusion

We have successfully constructed a method for converting CSG input into connected boundary representations. The method begins by taking the CSG primitives and constructing a contour boundary representation. We developed an algorithm for identifying the geometric concurrencies in a robust fashion, as well as provide stipulations on when user inputs become ill conditioned. The geometric concurrencies are replaced with topological concurrencies, and an intersection graph can be built using the location of boundary intersections and the ends of concurrencies. A sweep reduction algorithm was implemented which identifies which edges participate in various CSG operators, allowing complex boundaries to be constructed from the base boundaries. The algorithm requires no extra user input for constructing connected parts.

[1] Some intersections are too close to distinguish visually.

Simulation and Visualization of Particle Transport in Strongly Coupled Plasmas

By Caleb Levy and James Schloss, Jerome Daligault (Mentor, T-division)

Abstract

Understanding how correlations mediated by the Coulomb interactions between ions and electrons affect the transport properties of plasmas is interesting both from a basic physics standpoint and as a practical matter. Plasmas in several modern research areas, including high-energy density plasmas, astrophysical plasmas, ultracold plasmas and dusty plasmas can be strongly coupled. The primary focus of this project is to visualize molecular dynamics simulations of the one component plasma and Thomas-Fermi models in the strongly coupled warm dense matter regime by using the open source 3D modeling software, Blender. Visualization of this nature provides an educationally insightful and physically accurate depiction of molecular dynamics simulations. Visualization of how the individual and collective motion of particles in a plasma evolve from a nearly collisionless, gaseous regime continuously to an increasingly correlated, liquid-like regime as correlations among particles increase is achieved by visualizing the one component plasma model. This is possible by approximating the ions in the one component plasma model as spheres to be rendered by Blender at the appropriate Cartesian coordinates. Using Blender, the electron density of Thomas-Fermi molecular dynamics simulations of warm dense matter is also visualized. This is achieved by treating each point on the output density grid as a voxel to be rendered as a single voxel volume texture by Blender. The visualization interface we have developed this summer provides a unique physical depiction of dense, strongly coupled plasmas and will be exploited by our mentor in the future.

Introduction

Visualization of Molecular Dynamics (MD) simulations is currently possible through the use of software such as ParaView, VisIt, or Visual Molecular Dynamics. However, these visualization softwares tend to be designed to represent massive datasets adequately, and they are not necessarily intended to make engaging and professional looking visualizations in a scriptable and automated way, nor is animation a core concern for these packages. Making useful pictures in any scientific discipline is a difficult task, and this problem is magnified when making 3D images. Basing movies on these datasets which are both objectively useful and feasible for a typical scientist to make is harder still. For the purposes of our visualization, we used Blender, an open source, 3D modeling suite.

Though atypical for scientific visualization, Blender is fully python scriptable and capable of running without initializing the Blender user interface. For this reason, Blender is an ideal candidate for overnight rendering of visualized data. Blender also supports other necessary features for scientific visualization, such as: high quality images, video rendering, volume rendering with either point density or voxel textures, and a standard Cartesian coordinate syste m for the correct placement of particles or other types of information during visualization. With Blender's capabilities as image and video rendering software, scientific visualization with Blender has the potential to be incredibly high in quality, allowing for a more precise and accurate depiction of data; however, titles, color ramps, and other typical labels on data is cumbersome to create in Blender and much easier to use other software for, like imagemagick or ffmpeg. This means that visualization with Blender requires a comparably larger amount of scripting to recreate some features that are native in visualization software intended for scientific purposes.

In this report, we outline how we tackled the problem of visualizing One Component Plasmas (OCP) and Thomas Fermi Molecular Dynamics simulations by utilizing Blender. In visualizing the OCP model, we treated the ions as a sphere meshes, and use Blender's keyframe-based animation system for quickly creating clear and smooth videos of the plasmas. To be rendered at the appropriate Cartesian position by Blender and then interpolated between scenes to create videos with a high number of frames per second. In visualizing the Thomas-Fermi model, we made use Blender's volume render capabilities by translating our data into voxel grids, incorporating them as 3D textures. In doing we have established an extensible pipeline for creating high quality 3D animations of a variety of Molecular Dynamics simulations which give us insight into these physical systems and will guide future investigations of High Energy Density Physics. This gives us a foundation to build on for both scientific exploration of high energy plasma physics and creating intuitive and informative representations of this work.

Blender Scene Generation

For the purposes of python scripting, blender uses the python library "bpy." In addition, the native python version on blender is 3.4. To run a python script, "script.py", using the bpy library and the native Blender python version, run "blender -b -P script.py". The "b" flag initializes Blender without the user interface and the "P" flag immediately runs the python script.

To recreate standard settings for MD visualization we used the following settings: To create a standard black background, we set the "horizon color" to (0,0,0). In order to generate a "box" that surrounds the simulation, we created a cube that was the size of the MD simulation box, appended a "WIRE" material to it, used a color ramp to color it white, switched the diffuse shader to "FRESNEL," and set the diffuse intensity to 1.

Before rendering each video, we generated individual scenes in the form of PNG files in a separate directory and used imagemagick to place a timestamp and title on each image. In the case of Thomas-Fermi molecular dynamics visualization, where volume rendering was used, we also used Blender to create a color ramp and used imagemagick to append it on the side of each image. This provided labels on our visualization of simulations. A blank density scene is shown in figure 24.

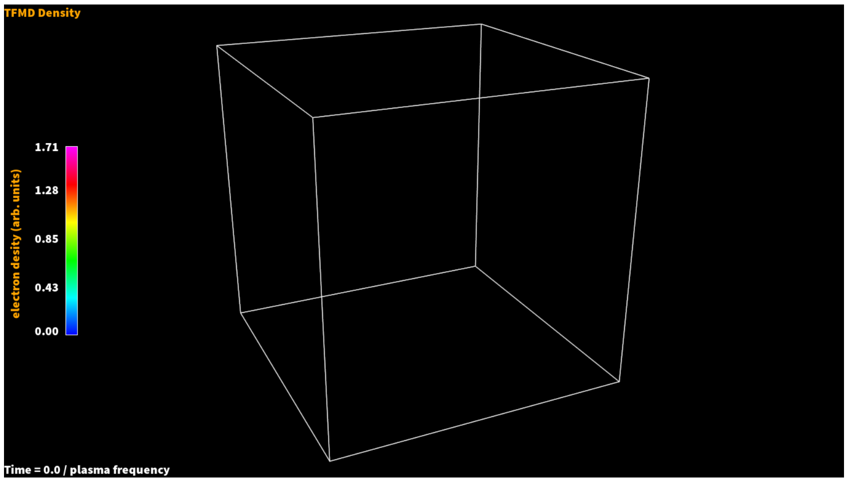


Figure 24: A single frame of an electron density plot without data visualized. This is the default scene for our visualization routines. Note the labels "TFMD Density" in the upper left, "electron density (arb. units)" beside the color ramp, and the timestamp in the lower left were generated by imagemagick using the wand python module. The color ramp was created with the blender pixel editor, but stamped onto the density plot with wand.

One Component Plasmas

In pursuing the creation of controlled fusion power it is necessary that we gain greater understanding of High Energy Density Physics, a discipline concerning itself with extreme states of matter occurring in thermodynamic regimes which do not normally occur on Earth. Among the relevant topics to this pursuit are non-ideal plasmas, for which the conventional framework of plasma physics does not apply.

Conventional plasma physics deals with high-temperature plasmas in a rarefied "gaseous" state. At higher temperatures and/or densities, plasmas begin to exhibit "liquid-like" behavior, characterized by the formation of structural order between ions of the plasma, similar to the short-range structures formed between atoms and molecules which distinguish normal liquids from ideal gases. These "liquid-like" plasmas fall under the domain of non-ideal plasmas.

Non-ideal plasmas are substantially more complicated than their ideal counterparts. Finding useful analytic solutions and approximations to the equations governing their behavior is an intractable challenge. Computational models of their behavior have thus been critical to understanding how they function. One of the key models for simulating non-ideal plasmas is the one-component plasma model.

A one-component plasma is a model used to describe the equilibrium dynamics of ions in dense, strongly coupled plasmas found in high energy-density experiments and compact astrophysical objects. This is a regime where the electrons are free and quantum mechanical and the ions interact classically as a single species of N point particles of charge Ze and mass m. For a Coulomb one-component plasma (OCP) the interaction potential of two particles separated by a distance r is given by the familiar Coulomb potential,

$$V(r) = \frac{(Ze)^2}{4\pi\varepsilon_0 r}.$$

For an OCP at equilibrium temperature T, the behavior of the plasma can be characterized by a dimensionless parameter: the coupling parameter,

$$\Gamma \equiv \frac{(Ze)^2}{4\pi\varepsilon_0}\frac{1}{ak_BT},$$

where $a = \left(\frac{3}{4\pi n}\right)^{1/3}$ is the average inter-particle distance, and n is the number density. This model is very handy because it is theoretically straightforward to describe and simulate; all relevant properties depend on single variable. The coupling parameter can be thought of as a ratio of Coulomb potential energy to kinetic energy.

For small Γ, the potential energy effects are very small, and the behavior is dominated by the thermal energy. In fact, a classical plasma is essentially one where $\Gamma \ll 1$. As Γ increases, the OCP will change from a nearly collisionless gaseous regime into an increasingly correlated, dense, liquid-like regime. A value of $\Gamma \geq 1$ represents a non-ideal, 'strongly coupled' plasma. For even higher Γ, when $\Gamma \geq 175$, the free energy of the fluid phase matches the free energy of the solid phase, at which point the plasma crystallizes. The regimes of interest in this paper are for values of coupling parameter less than the crystallization value but greater than for the gaseous state.

One of the most important diagnostics of these simulations is the pair-distribution function, $g(r)$, defined as the average density of a plasma as a function of radius from any given particle. For small Γ, when the plasma is in a gaseous state, the pair-distribution function is uniform, as there is no structure to the particles. As Γ increases, we see peaks forming in $g(r)$, corresponding to small scale irregular structure, thought to be from the formation of fluid parcels. Finally, as the plasma solidifies at very high $g(r)$, we see spikes corresponding the regular spacing from a semi-crystalline structure.[44] Plots of the pair-distribution function can be seen in figure 25.

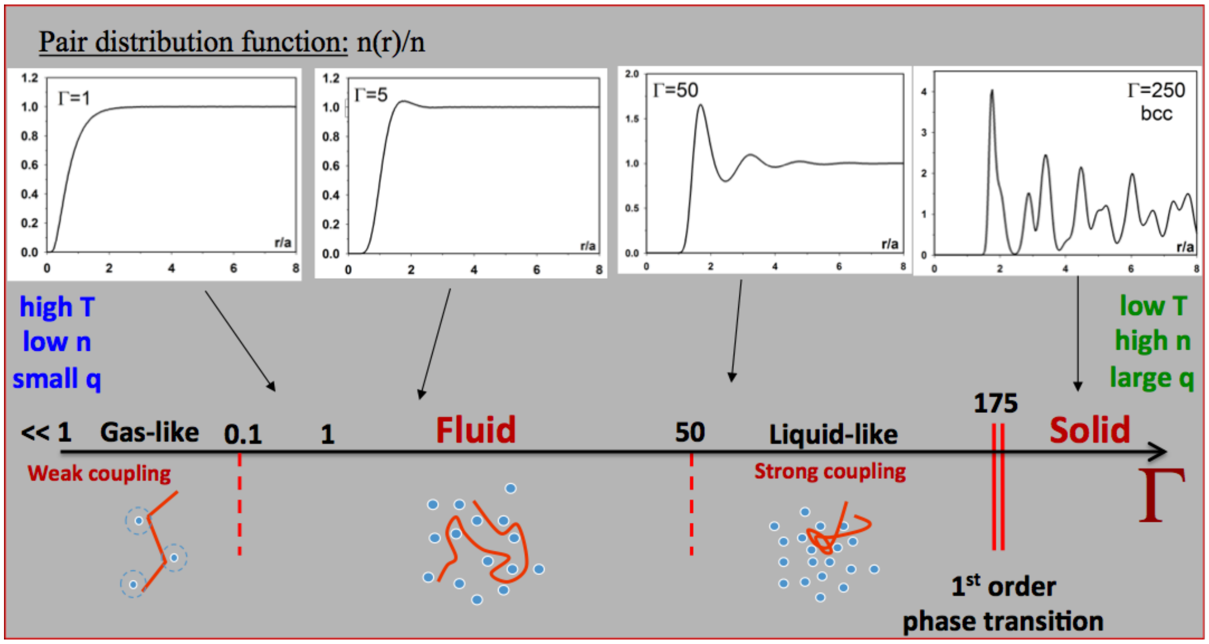


Figure 25: Plots of the pair-distribution function for increasing values of Γ, placed along an axis corresponding the plasmas state. Image courtesy of Jerome Daligault, from a talk given to the University of Iowa in 2014.[14]

This summer, we wanted to make new visualization software to gain insight and intuition into the behavior of these plasmas. Although the existing diagnostics are useful in determining the state of the plasma at any given time, we wanted to explore more qualitative aspects of their behavior. In particular, we were interested in studying a behavior termed “caging,” a fluid-like plasma phase occurring at medium values of gamma, where the particles are believed to trap each other into fluid parcels which form the beginning stages of when a continuum approximation becomes accurate. To do this we made use of Blender’s visualization and animation capabilities.

Blender was appropriate to this investigation for a number of reasons. Its interface allows for easy interaction with the particles, allowing one to fly, pan and zoom in and out of a box of particles, while animating their trajectories in real time. This can be seen in figure 29. The viewport uses a “preview” render, and thus is not meant for creation of production quality videos. However importing data into the Viewport takes on the order of seconds, and real-time interaction with the data is very useful in gaining quick intuition about how the plasma behaves.

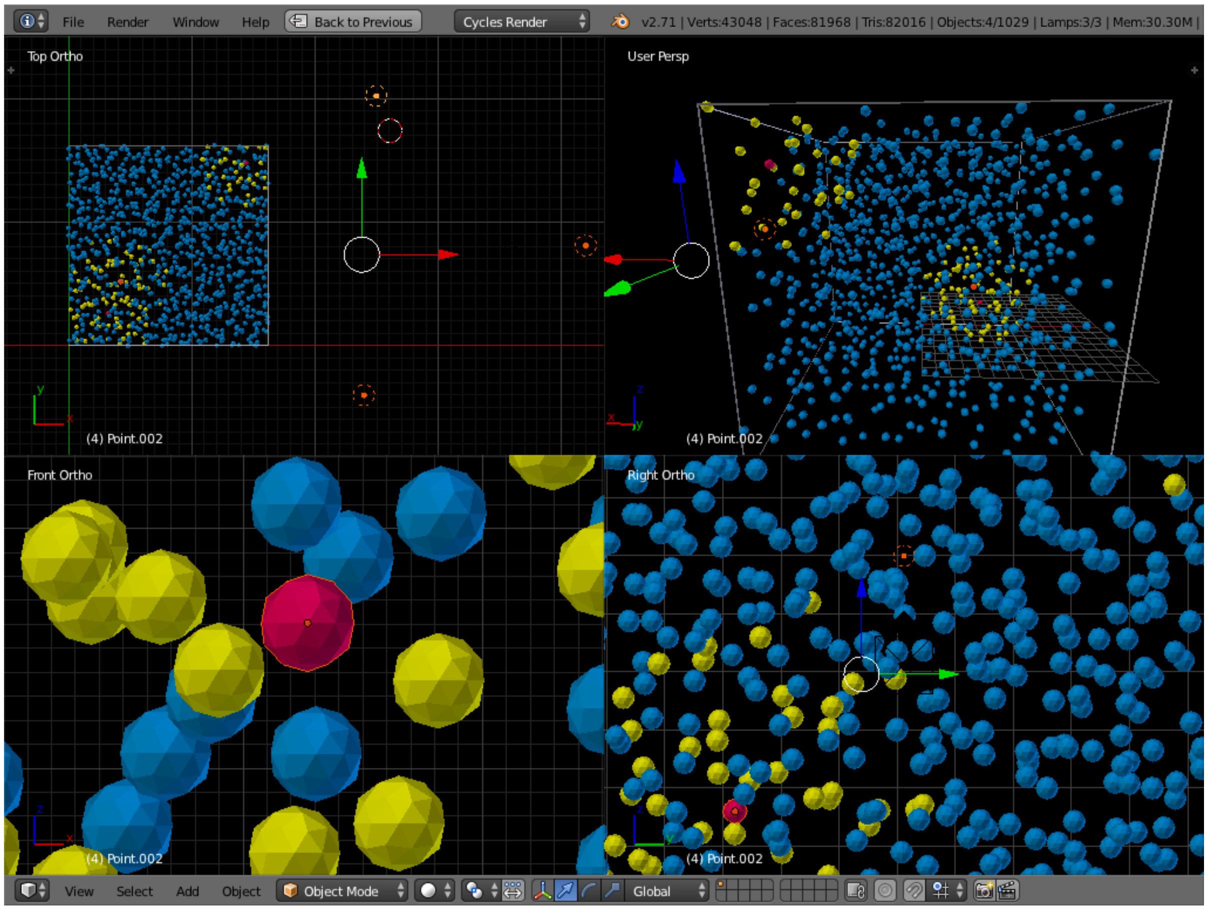


Figure 26: Snapshot of Blender's "viewport," the interaction window for the application, used to set up the scene for final render. In this image the viewport is set in quad-view: the top-left, lower left and lower right panes depict orthographic projections of the scene onto the xy, xz and yz planes, respectively, and the fourth panel is shown in "camera view," the perspective from which the final render will take place. Also present are "tracer particles," in red, along with their neighbors in yellow. As the animation plays we can see how exactly the tracers interact with their neighbors, gaining insight into the phenomenon known as caging.

Blender's ability to create real time previews of the rendered movies comes from it's approach to depicting motion. Instead of recreating each mesh at a new location at each point in time, we can create a single mesh in blender, and assign "keyframes," or "fcurves" to that mesh. These keyframes describe where an object needs to be at a given point in time.[2] From this data, blender will simply interpolate the motion of the objects between these keyframes, and create the animations on the fly. Blender's animation interface is extremely powerful, and extends well beyond moving objects around. Fcurves are represented as a very simple sequence of time-value pairs, and due to their simplicity they are quite general. A user can keyframe such data as the color or an object, the focal length of a lens, object visibility and transparency, and even the vertices of a mesh. Quite literally, just about any adjustable property of blender's

[2]In blender time is, for the most part, measured in the frames of a video, although the methods of interpolation between different frames can be modified.

interface can be keyframed and animated.

For the one component plasma, in order to perform actual renders of the particles, we used the cycles render engine. Cycles is a ray-tracer,[3] which essentially shines light-rays onto the scene using data from the existing meshes, and follows their trajectories to produce an image. By placing artificial "lamps" and "suns" into the scene, we can create finely controlled and realistic light schemes as a natural byproduct of how blender renders an image. Additionally, all of the typical "Hollywood techniques" for changing an object's texture and look, along with control of camera focal length, position and rotation, are available for use in these animations. These capabilities are of great utility making interesting and informative depictions of the OCP. Demonstrations of these abilities can be seen in figure 27.

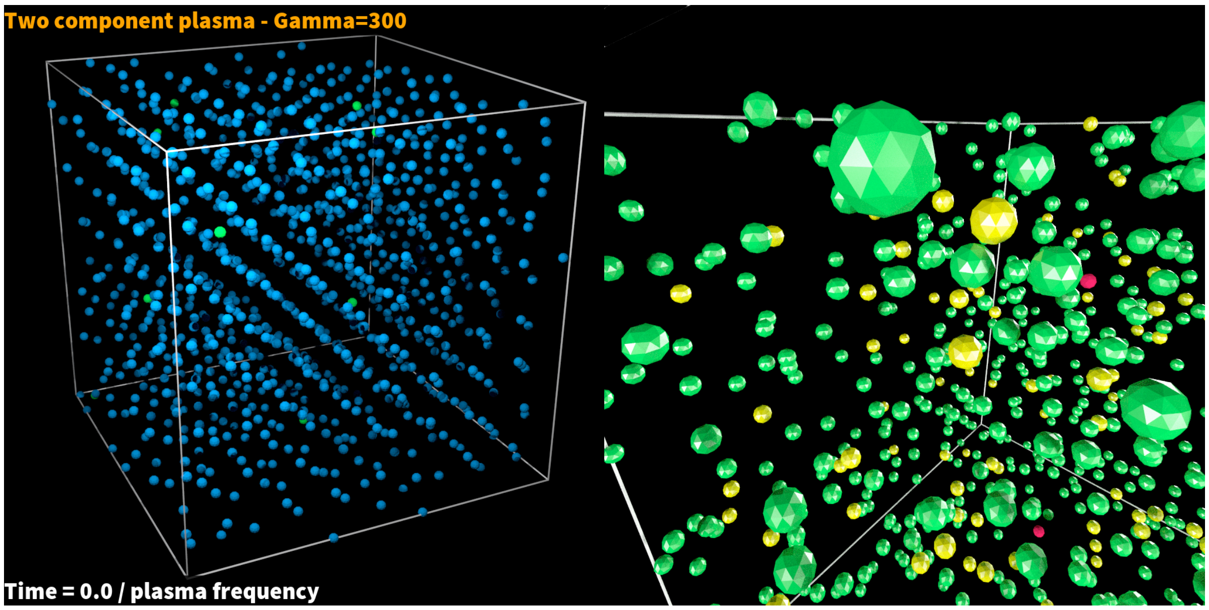

Two component plasma - Gamma=300
Time = 0.0 / plasma frequency

Figure 27: A demonstration of using very different representations for the same (or similar) data using blender. On the left the camera is placed some distance away from the box of particles and then the lens is zoomed in, creating a relatively flat scene. Here the lights are concentrated toward the top corner of the box, giving us a sense of the depth of the particles based on how bright or dim they are (i.e. vicinity to the light source). On the right we have used a more metallic material to make the particles more shiny and reflective, and thus lighter. We have also moved the camera towards the top of the box, and then zoomed the lens out to gain a wider field of view. All of the particles in the right-hand box are evenly lit, but due to the wide-angle lens we can rely on the parallax caused by perspective distortion to give us a sense of the particles' distance from the viewer. Both of these techniques are meant to address one of the hardest problems with making 3D visualizations using 2D images: conveying a sense of depth.

To make this code more easily usable, we created a wrapper script called "make_movie,"

[3]In particular, cycles is a path tracer, a particular kind of ray tracer which directs beams outward from the camera and onto the scene until they hit a light source. Regular ray-tracers direct beams outward from the light sources until they hit the camera, essentially taking the opposite approach. The operational principles behind ray-tracers and path tracers are identical, however.

which will process the text file containing particles' positions, turn them into a usable format, set up the scene, camera angles and lighting, and add all the particles and their trajectories. Using this script we can adjust the output movie formats, effective frame rates and resolutions. We use a parameters file to add tracer particles at select locations as well. The modules are built to be customizable and expandable. For example, right now camera angles are chosen from a selection of preset values, but it would be relatively straightforward to implement a general camera setting. Similarly, while currently particle colors are hard coded in, it would be quite straightforward to make them user settable.

These animations proved extremely helpful in our investigation of the OCP behavior. Because we could display multiple animations simultaneously, we could compare the evolution and behavior of the particles in different states of matter. This was particularly helpful for understanding a two component plasma, where different species of particle existed in different states of matter at the same time. These movies were presented at the 2014 international conference for Strongly Coupled Coulomb Systems, held in Santa Fe, NM. Snapshots of these animations can be seen in figure 28.

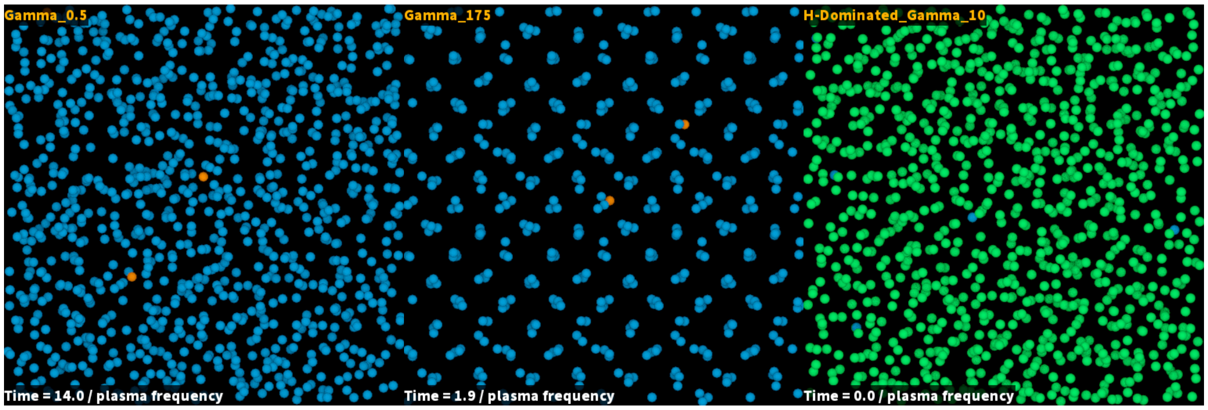

Gamma_0.5
Gamma_175
H-Dominated_Gamma_10
Time = 14.0 / plasma frequency
Time = 1.9 / plasma frequency
Time = 0.0 / plasma frequency

Figure 28: Snapshots from three movies featured at the International Conference for Strongly Coupled Coulomb Plasmas. To the left is a one component plasma with $\Gamma = 0.5$, in a gaseous-like state. Shown in the middle is a plasma with $\Gamma = 175$, already in a crystalline state. The orange particles in the first two snapshots are "tracer particles." At $t = 0$ these tracers will be in the same position for both $\Gamma = 175$ and $\Gamma = 0.5$. One can play both movies simultaneously to see that the tracers in the gas quickly spread apart while tracers in the crystalline solid simply oscillate about their positions in the lattice. On the right is a two component plasma, with iron ions in green and hydrogen in blue. When playing the animations, one can clearly see the Fe ions behave as a liquid, and the H ions behave like particles in a gas, demonstrating the existence of two different simultaneous states of matter.

Finally, our preliminary investigations of caging indicate that we may need to change our description of the phenomenon. The results so far indicate that at medium to high values of Γ a particle's neighbors will tend to diffuse away from it, as opposed to simply trapping it in place. This can be seen in the snapshots of figure 29, where the particles can be seen spreading apart as time progresses. It is an important discovery that the previous descriptions were, at least partially, incorrect, and it is the sort of finding which would have been very difficult without visualizations of this nature.

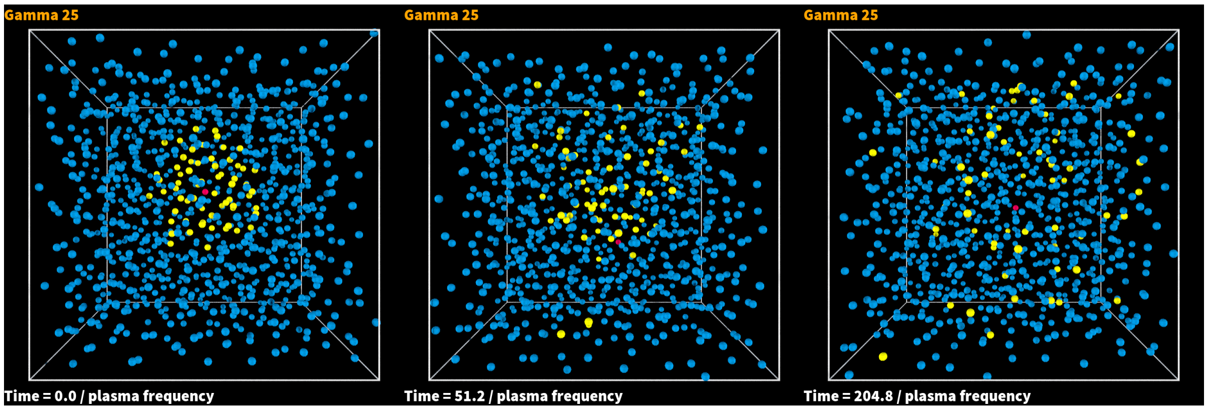

Gamma 25
Gamma 25
Gamma 25
Time = 0.0 / plasma frequency
Time = 51.2 / plasma frequency
Time = 204.8 / plasma frequency

Figure 29: Snapshots from an animation of an OCP plasma at $\Gamma = 25$ meant for visualizing "caging." At $t = 0$, the closest particles to the center (red dot) at colored yellow. As time progresses in the animation we can see these neighbors disperse. By $t = 50$ inverse plasma frequencies, we can see that the neighbors have started to diffuse away from the central particle, losing most of their structure. By the end of the simulation at $t = 200$ inverse plasma frequencies, The initial neighbors of the central particle have completely diffused away with little resemblance to their initial state. This is somewhat different from the expected behavior of particles "trapping" each other into fluid-like parcels.

Thomas Fermi Molecular Dynamics Visualization

Our visualization of Thomas Fermi Molecular Dynamics simulations is motivated by the pressing need to understand and predict the basic properties of matter in the warm dense matter regime [51]. The warm dense matter regime encompasses ionized fluids at the confluence of condensed matter physics, plasma physics, and dense liquids. Warm dense matter is generated in many laboratory experiments that start from a cold sample, including laser-produced plasmas and the implosion phase of inertial confinement fusion capsules. Warm dense matter is also at the heart of numerous unsolved problems in planetary science and stellar astrophysics. Roughly speaking, and depending on the material, warm dense matter conditions occur for temperatures 0.5 to 100 eV (1eV = 11604 Kelvin) and densities 0.1 to 50 times the normal solid density. Under these conditions, matter is a partially ionized fluid with significant temperature effects, where quantum mechanics and statistics govern the behavior of the electrons. Ions are classical, but in a moderately to strongly correlated, liquid-like state. Understanding how the interplay of correlation and quantum effects gives rise to the physical properties of warm dense matter is a frontier of high energy density physics.

Experimental measurements in the warm dense matter regime are conspicuously sparse and computer simulations play the leading role in today's investigations of this extreme state of matter. Visualization of warm dense matter simulations aim at accurately describing the behavior of a large number of ions and electrons with periodic boundary conditions.

In the prevalent simulation tools for warm dense matter used today, ions propagate classically according to the orbital-based, Kohn-Sham implementation of density functional theory at finite temperature [37]. Kohn-Sham density functional theory is currently the most widely used technique in many branches of material science and chemistry [5]. While the majority of calculations in these fields focus on ground-state properties, i.e. at zero temperature, similar calculations in the warm dense matter regime are vastly more computationally expensive. For this reason, we have decided to use the Thomas-Fermi approach, which is much less expensive for warm dense matter regime simulations.

Below we give the main equations that our numerical implementation must solve. While our mentor's molecular dynamics code will account for several chemical species (mixtures are important to warm dense matter research), for clarity the expressions given below assume a single species of ions. Our mentor models a warm dense matter system as a neutral, nonrelativistic system of N ions (mass M, charge Z, position R_i) and of $N_e = ZN$ electrons (mass m, charge $-e$). The force on an ion i is:

$$M\frac{d^2\mathbf{R}_i}{dt^2} = Ze^2\sum_{k\neq i}^{N}\frac{\mathbf{R}_i-\mathbf{R}_k}{|\mathbf{R}_i-\mathbf{R}_k|^3} - \frac{\partial}{\partial\mathbf{R}_i}F[n_{eq},\mathbf{R}] \tag{6}$$

where the first term describes the Coulomb interactions between ions, and the second term describes the force on the ions due to their interaction with the electrons. Here we have indicates explicitly that the free-energy functional, $F[n,\mathbf{R}]$, is both a functional of the electron density and a function of all ionic positions. Hence, given a set of ionic positions at some time, the force pushing the ions requires the calculation of the equilibrium electronic density in this fixed ionic configuration. The free energy is conveniently expressed as the sum:

$$F[n,\mathbf{R}] = F_0[n] + \frac{e^2}{2}\int d\mathbf{r}n(\mathbf{r})v(\mathbf{r}) + F_{xc}[n] + \sum_{j=1}^{N}\int d\mathbf{r}v_{ie}(\mathbf{r}-\mathbf{R}_j)n(\mathbf{r}) \tag{7}$$

The first term in the sum corresponds to the free-energy of a non-interacting electron system. The second term describes the interaction with their own Coulomb mean-field v($\mathbf{r}$), which is a solution of Poisson's equation $\nabla^2 v(\mathbf{r}) = n(r)$. In our mentor's implementation, the latter is solved on a uniform grid using 3D fast Fourier transforms. The third term, $F_{sc}[n]$, describes the so-called exchange-correlation corrections not accounted for by the first two terms. Accurate approximations in terms of density only have been developed for both zero and finite-temperature systems [34]. The fourth term describes the electron-ion interaction through $v_{ie}(r)$. The only difference between the KS method and the orbital-free method lies in the first term, $F_0[n]$.

These numerical algorithms were implemented in a massively parallel molecular dynamics code developed by our mentor over the past years. The code combines high-resolution of close encounters (which are dealt with using nearest neighbor techniques) and rapid, long-range force calculations (which are computed on a mesh with 3D fast Fourier transforms). Currently, the prevalent implementations of orbital-free density functional theory for warm dense matter studies use the finite-temperature Thomas-Fermi approximation [19], the simplest of all orbital-free approximations for $F_0[n]$.

Visualization of Thomas-Fermi Molecular Dynamics with Blender

Unlike one-component plasma visualization, Thomas-Fermi molecular dynamics visualization must take electron density into account and thus requires volume rendering techniques. Blender has a native volume texture and two methods to color the volume, both of which require appending objects with a “volume” material and unique texture. The first method is named “point density.” This technique allows the user to implement a particle system inside a volume and color that volume based on the individual particle’s velocity, speed, or time alive. Though this technique does not currently have the capabilities to color a volume rendered cube based on spatial location, it may be added in the future, which would allow for easier modeling of point systems with Blender.

The second volumization technique is known as “voxel data.” This system allows the user to input a standard 8-bit RAW voxel file to be plotted in a cubic volume by Blender. This is the technique we use to visualize Thomas-Fermi molecular dynamics simulations. In order to do so, we output the data of the particles in the simulation and the densities at each point in a 64 x 64 x 64 voxel grid, where each voxel point holds the normalized density data for that location in the MD cube. The data is organized to cycle through the voxels in each individual x column before moving to the next y column. Once a plane of data points is created, the data is organized to move to the next z column. In this way, a 3D voxel cube is created. This cube is then resized to match the appropriate box length for the simulation and moved to the appropriate location. Using Blender’s python interface, we create a color ramp to appropriately visualize the internal data of the molecular dynamics cube, where varying levels of electron density also vary the color in that region of the molecular dynamics cube. If spheres are rendered (as in the one-component plasma visualization above) to visualize particles, we expect to see areas of high electron density surrounding the particles (figure 30).

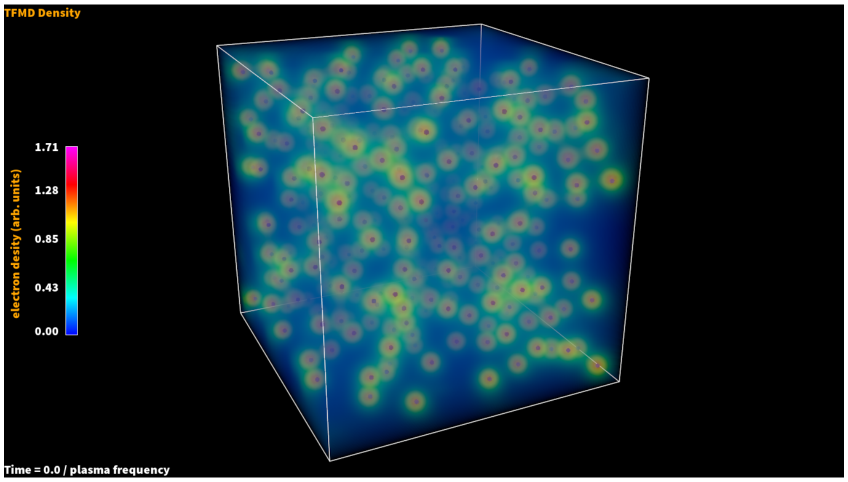


Figure 30: A single frame of an electron density plot with data visualized. Note that the areas of high electron density are surrounding the red particles. This simulation was run with 300 Boron ions in a 5eV plasma, 10g/cc

Additional Features

Due to the nature of volume rendering, some data will be occluded in the center of the cube. Though the Blender color ramp allows users to set an appropriate “alpha” value, the transparency of an object, it is still insightful to take two-dimensional slices to visualize data at specific regions in the interior of the molecular dynamics cube. For this reason, we allow the user to set a variable location on the x, y, and z axis to “slice” the Thomas-Fermi molecular dynamics cube. Due to the method of organizing our data, we are able to create additional voxel grids that carry only a single plane of data at a region specified by the user. For example, a 2-dimensional slice along the y, z plane of the Thomas-Fermi molecular dynamics cube at the center could be created by entering a value of half the box-length for the x value. This would create a 1 x 64 x 64 voxel grid at the x column location closest to the value entered. Using the above procedure, the voxel grid is then appended to the volume texture of another cube, and the cube is resized to appear as a y, z plane. Three slices, one for each cartesian plane, are shown in figure 31.

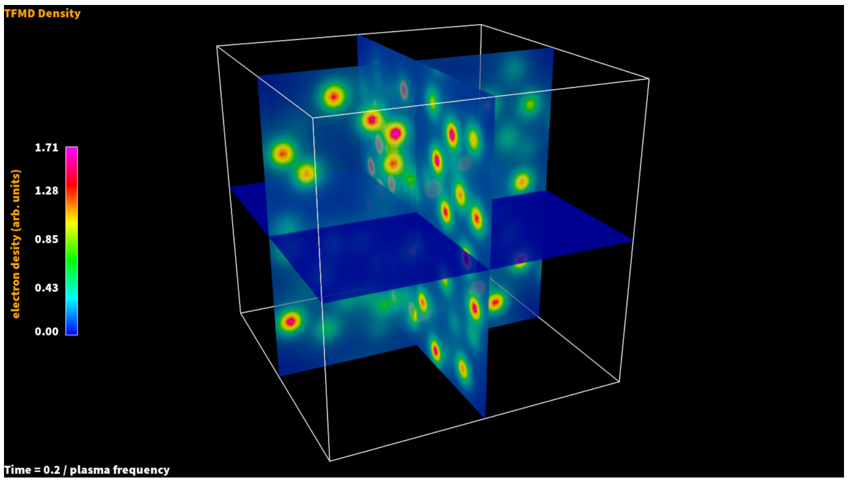


Figure 31: Three slices, one for each cartesian plane, of a Thomas-Fermi molecular dynamics visualized cube. This simulation was run with 300 Boron ions in a 5eV plasma, 10g/cc

In addition to the 2-dimensional slices, we also incorporate 9 unique and standard color schemes (figure 32), each of which utilize different color ranges of Blender’s color ramp. We also include a number of standard camera angles (figure 33). These features should enable users to visualize their data in a standard and physically accurate way. Of course, new camera angles and color schemes can be implemented by setting the desired parameters.

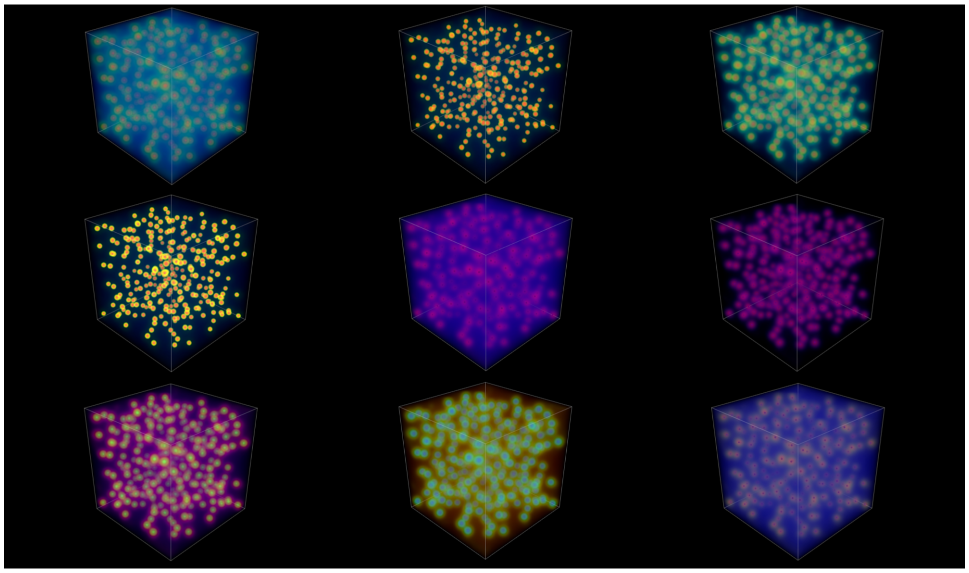


Figure 32: The incorporated color schemes in the initial release of our visualization method. These simulations were run with 300 Boron ions in a 5eV plasma, 10g/cc

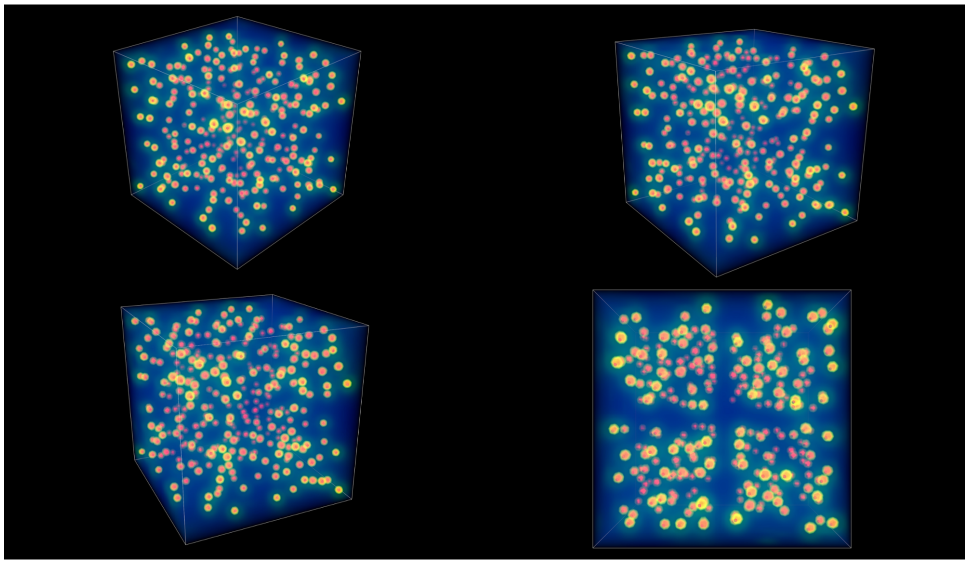


Figure 33: The standard camera angles incorporated in the initial release of our visualization method. These simulations were run with 300 Boron ions in a 5eV plasma, 10g/cc

Final Product

This summer we formed a software foundation on which to build for future work. As described earlier, blender was not built with scientific visualization in mind, and thus requires a fair amount more work to incorporate it into a usable workflow. Much of what we have done this summer is just setting that pipeline up. We intend to continue forward with the software we have built to make greater use of blender's powerful visualization capabilities.

For the one component plasma model, we intend to incorporate such features as adding trails to better see the trajectories of the particle trajectories and determine how the motion evolves. We are also contemplating using keyframes to animate color transitions and alpha transparencies to particles to help accentuate the depictions of particle cages. Finally we wish to use this software to explore multi-component plasmas, an application for which it is particularly well suited by comparison to existing diagnostics and techniques.

For our TFMD visualization routine, we would like to look into interpolating between scenes and points to create smoother videos and images. We would also like to look into modularizing the script so we can create movies and images directly from our mentor's MD code without first creating a data file. This would allow us to increase the efficiency of our code and use a much smaller amount of storage. In addition, we would like to look into different color ramp scaling schemes and

For both the OCP and TFMD visualization scripts, we would like to add some more advanced functionality typically employed in commercial entertainment movies. These include camera panning and zooming, and using multiple views of the data spliced together. In the same vein, we wish to take advantage of changing lighting dynamically to emphasize certain temporal phenomena. We may also explore the possibility of making stereoscopic three-dimensional movies; i.e. using 3D glasses with our animations.

Finally, as it stands the visualization codes for the OCP and TFMD simulations are currently separate. In the future we wish to combine both codes to be used with greater effectiveness together, both for ease of use and more effective visualization of the interactions of ions with their electron clouds.

Computation of the Ion-Ion Dynamic Structure Factor in Warm Dense Matter

By N. Matthew Gill and Robin A. Heinonen
Didier Saumon and Charlie Starrett, mentors

Abstract

Using the pseudo-atom molecular dynamics (PAMD) model of Starrett and Saumon, the ion-ion dynamical structure factor is computed using classical molecular dynamics simulations. Results for warm dense aluminum are compared to previous results obtained from Kohn-Sham density-functional theory and orbital-free density-functional theory, and the first simulations of warm dense mixtures using a realistic model are performed, in which we search for Fano resonances.

Introduction

Warm dense matter (WDM) is an intermediate regime that bridges the gap between the solid state and classical plasmas. It refers to plasmas at roughly solid density and temperatures on the order of 1 to 100 eV [65]; the regime is indicated in the phase diagram of Fig. 34. The ions in WDM are strongly coupled, and the electrons are partially degenerate and behave quantum-mechanically; as a result, WDM is difficult to model [44, 59, 71, 65]. WDM occurs in a number of astrophysical contexts (including the cores of Jupiter-like planets and dense stars), as well as inertial confinement fusion (ICF) experiments [59, 71, 65].

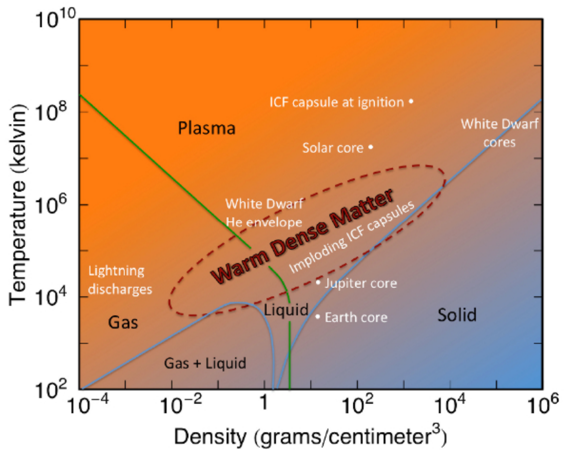


Figure 34: Phase diagram indicating the WDM regime.

An experimentally accessible quantity known as the dynamic structure factor (DSF) encodes profound statistical and thermodynamic information about WDM. X-ray Thompson scattering (XRTS) experiments, which uses narrow-bandwidth x-ray laser sources as a probe of WDM [23, 65], have only very recently resolved the narrowest feature of the WDM DSF, the ion-ion component. Models for this feature tend to suffer either from high computational expense or excessive approximation. However, a model developed by Starrett and Saumon [67, 68], "pseudo-atom molecular dynamics" (PAMD), allows the ions in WDM to be accurately simulated using classical molecular dynamics (MD) simulations, from which the ion-ion DSF may be computed rapidly. In this paper we discuss the computation of the ion-ion feature and compare the results for warm dense aluminum to those of previous models, before turning our attention to a novel arena of study: warm dense mixtures, in which a recent paper [63] predicted the occurrence of Fano resonances in the DSF.

Theory

The dynamic structure factor

In XRTS experiments, the double differential cross section is directly proportional to the dynamic structure factor $S(k,\omega)$:

$$\frac{d^2\sigma}{d\omega_0 d\Omega} = |\boldsymbol{\varepsilon_0}\cdot\boldsymbol{\varepsilon_1}|^2 \left(\frac{e^2}{m_e c^2}\right)^2 \frac{\omega_1}{\omega_0} S(k,\omega), \tag{8}$$

where $\omega = \omega_0 - \omega_1$ is the energy transfer, $\mathbf{k} = \mathbf{k_0} - \mathbf{k_1}$ is the momentum transfer, $\boldsymbol{\varepsilon_1}$ and $\boldsymbol{\varepsilon_1}$ are (respectively) the polarizations of the incoming and outgoing x-rays, and $\hbar$ has been set to unity.

The DSF is also related to the linear response function $\chi(k,\omega)$ via the fluctuation-dissipation theorem [44]:

$$S(k,\omega) = -\frac{k_B T}{\pi n \omega}\Im(\chi(k,\omega)), \tag{9}$$

where n is the number density. Hence, beyond its importance in experimental contexts, $S(k,\omega)$ contains statistical information as well. Furthermore, it may be used to compute thermodynamic quantities such as the pressure [22].

Chihara's formula [10] expresses $S(k,\omega)$ as the sum of three terms. Explicitly,

$$S(k,\omega) = \underbrace{|f(k)+q(k)|^2 S_{ii}(k,\omega)}_{\text{ion-ion}} + \underbrace{\bar{Z}S_{ee}(k,\omega)}_{\text{free-free}} + \underbrace{S_{bf}(k,\omega)}_{\text{bound-free}}, \tag{10}$$

where $f(k)$ and $q(k)$ are, respectively, the Fourier transforms of the the ion and screening electron densities, and $\bar{Z}$ is the average ionization. The free-free and bound-free features have been modeled in the PAMD framework previously [65]. In this paper, we compute the narrower ion-ion feature, which has a bandwidth on the order of 0.1 eV [59] and which has recently been measured for the first time (presented at Strongly Coupled Coulomb Systems 2014).

Mathematically, the function $S_{ii}(k,\omega)$ (which we henceforth refer to as $S(k,\omega)$ and the DSF without ambiguity) is the Fourier transform of the ion density-density correlation function [27]. In general, the microscopic ion density is given by

$$\rho_{\mathbf{r}}^{\mu}(t) = \sum_{i=1}^{N^\mu} \delta(\mathbf{r} - \mathbf{R}_i(t)), \tag{11}$$

where N^μ is the number of particles of species μ and $\mathbf{R}_i(t)$ is the location of the ith particle. The Fourier component of the density is then

$$\rho^\mu_{\mathbf{k}}(t) = \sum_{i=1}^{N^\mu} \exp(i\mathbf{k}\cdot\mathbf{R}_i(t)) \tag{12}$$

The (partial) *intermediate scattering function* (ISF) is then defined as

$$F^{\mu\nu}(k,t) = \frac{1}{\sqrt{N^\mu N^\nu}} \langle \rho^\mu_{\mathbf{k}}(t)\rho^\nu_{-\mathbf{k}}(0)\rangle, \tag{13}$$

where the average is taken over all time origins and all wavevectors $\mathbf{k}$ such that $|\mathbf{k}| = k$ (under isotropic conditions, $F^{\mu\nu}$ is in principle independent of the direction of $\mathbf{k}$).

The partial dynamic structure factors, finally, are the time Fourier transforms of these partial ISFs:

$$S^{\mu\nu}(k,\omega) = \frac{1}{\pi}\int_0^\infty F^{\mu\nu}(k,t)e^{i\omega t}\,dt. \tag{14}$$

The full DSF is just the sum of these partial DSFs.

Other correlation functions—the static structure factor and the longitudinal current correlation function

It is necessary to introduce two other, related correlation functions. The *static structure factor* $S(k)$ (SSF) is a time-averaged version of the dynamic structure factor. For species μ and ν we have the partial SSF

$$S^{\mu\nu}(k) = \frac{1}{\sqrt{N^\mu N^\nu}} \langle \rho^\mu_{\mathbf{k}}\rho^\nu_{-\mathbf{k}}\rangle, \tag{15}$$

where here the average is taken over all appropriate wavevectors $\mathbf{k}$ and over all time. The full SSF is, again, the sum of these partial SSFs. It is clear from the definitions that

$$F^{\mu\nu}(k,0) = S^{\mu\nu}(k). \tag{16}$$

We also have the sum rule [27]

$$S^{\mu\nu}(k) = \int_{-\infty}^{\infty} S^{\mu\nu}(k,\omega)\,d\omega = 2\int_0^\infty S^{\mu\nu}(k,\omega)\,d\omega. \tag{17}$$

Another function of interest is the *longitudinal current correlation function* (LCCF). In a given WDM fluid, the longitudinal current is given by

$$j^\mu_{\mathbf{k},\ell}(t) = \sum_{i=1}^{N^\mu} \frac{\mathbf{v}_i(t)\cdot\mathbf{k}}{k}\exp(i\mathbf{k}\cdot\mathbf{R}_i(t)). \tag{18}$$

We then have the partial LCCFs

$$C^{\mu\nu}_\ell(k,t) = \frac{k^2}{\sqrt{N^\mu N^\nu}} \langle j^\mu_{\mathbf{k},\ell}(t)\, j^\nu_{\mathbf{k},\ell}(0)^\star\rangle \tag{19}$$

where the average is over appropriate wavevectors **k** and all time origins, and the star denotes the complex conjugate. It can be shown that the LCCF is closely related to the DSF:

$$C_\ell^{\mu\nu}(k,\omega) = \omega^2 S^{\mu\nu}(k,\omega), \tag{20}$$

where $C_\ell^{\mu\nu}(k,\omega)$ is the power spectrum

$$C_\ell^{\mu\nu}(k,\omega) = \frac{1}{\pi}\int_0^\infty C_\ell^{\mu\nu}(k,t)e^{i\omega t}\,dt. \tag{21}$$

Further, for one-component WDM (not a mixture), the initial value of the time-domain LCCF takes the simple analytic form

$$C_\ell(k,0) = \frac{k_B T}{m}k^2, \tag{22}$$

where m is the ion mass. This will provide a useful check when computing the LCCF.

Pseudo-atom molecular dynamics

We now present a brief overview of the PAMD model. PAMD, a density-functional theoretic approach, uses a two-component plasma (TCP) model combined with an average atom model. In an average atom model, the superposition approximation for the electron density

$$n_e(\mathbf{r}) = \sum_i n_e^{PA}(|\mathbf{r}-\mathbf{R}_i|), \tag{23}$$

is made, where n_e^{PA} is called the pseudo-atom electron density. A spherically symmetric electron density is found by minimizing the free energy in a system consisting of a central nuclear charge surrounded by a spherically symmetric charge distribution found by spherically averaging the other ions. The electron density for the same system, but without the central nuclear charge, is also found and subtracted in order to determine n_e^{PA}. In the case of PAMD, the ion pair distribution function $g(r)$ (which is needed for the average atom model) is first approximated by the Heaviside step function $\Theta(r-R)$, where R is a function of the plasma density, and this is used to find the pseudo-atom electron density. The bound electrons of the pseudo-atom are then determined and their density is subtracted from the pseudo-atom density to obtain the screening electron density. Next, the Ornstein-Zernicke (OZ) integral equations are used to obtain an expression for the ion pair interaction potential in terms of the screening electron density. This potential, then, may be fed into classical MD simulations to obtain the ion configurations, whence at last we obtain the electron density via Eq. (23).

In principle, the coupling of the OZ equations with the average atom model provides a schema to self-consistently determine n_e^{PA} via iteration between the two models, but in practice the pseudo-atom electron density is insensitive to $g(r)$ and further iteration is unnecessary.

PAMD, since it utilizes classical MD with pair interaction potentials, has the key advantage of being much faster than the method of quantum molecular dynamics (in which one must at each time step minimize the free energy before calculating the forces on the ions), allowing for relatively rapid simulations on the order of thousands of particles and tens of thousands of time steps. Moreover, PAMD does not suffer from certain approximations used in other models, such as a pseudopotential.

Calculation and Verification

In order to calculate $S^{\mu\nu}(k,\omega)$ and $C_\ell^{\mu\nu}(k,\omega)$, we wrote a code in the FORTRAN 90 language and parallelized portions of the calculation using OpenMP. The code calculates all possible wavevectors $\mathbf{k}$ allowed by the boundary conditions of an MD simulation, and uses position and momentum information from the MD simulation to calculate the DSF and LCCF. Block averaging of the calculations from several different MD simulations was used in order to gain more consistent results and average out some of the statistical noise. After the block averaging, an automated noise smoothing routine was applied to the data in order to remove the non-physical noise in the correlation functions that developed at large values of time. In particular, the data was multiplied by the window function

$$w(t) = \begin{cases} 1 & \text{if } t < t_0 \\ \exp\left(-a\left|\frac{f'(t_0)}{f(t_0)}\right|(t-t_0)\right) & \text{if } t \geq t_0 \end{cases}, \tag{24}$$

where t_0 is the location of the first peak in the correlation function such that the following peak has a larger magnitude, f is the unsmoothed correlation function, and a is a positive real parameter chosen on a case-by-case basis (usually around 50). The derivative $f'(t_0)$ was determined using a two-point difference method.

The molecular dynamics code was written by Dr. Starrett prior to the commencement of the workshop. The MD simulations only required the input of ion-ion pair potentials generated from the PAMD model. Using the resulting position and momentum data, calculations of the relevant correlation functions can be carried out in a straightforward manner. In order to calculate the DSF, the following algorithm was applied (where calculations were carried out according to the mathematical definitions of the functions as outlined in the Theory section):

1. Using the particle position data, calculate the spatial Fourier transform of the ion densities, $\rho_{\mathbf{k}}^{\mu}(t)$, for every value of $\mathbf{k}$ and t

2. Calculate the autocorrelation of the Fourier space densities (the intermediate scattering function), $F^{\mu\nu}(\mathbf{k},t)$

3. Average $F^{\mu\nu}(\mathbf{k},(t)$ over all wavevectors, $\mathbf{k}$, having equivalent wavenumber k and over all time origins in order to obtain $F(k,t)$

4. Take the time Fourier transform of $F^{\mu\nu}(k,t)$ in order to obtain the dynamical structure factor, $S^{\mu\nu}(k,\omega)$. A trapezoid method was used to avoid a constant offset error

A direct calculation of the LCCF, $C_\ell^{\mu\nu}(k,\omega)$, was also carried out in a similar manner according to the following algorithm:

1. Using the particle position and momentum data, calculate the spatial Fourier transform of the ion current densities, $j_{\mathbf{k}}^{\mu}(t)$, for every value of $\mathbf{k}$ and t

2. Calculate the autocorrelation of the projection of the Fourier space current densities along the $\hat{k}$ direction, $C_\ell^{\mu\nu}(\mathbf{k},t)$

3. Average $C_\ell^{\mu\nu}(\mathbf{k},t)$ over all wavevectors, $\mathbf{k}$, having equivalent wavenumber k and over all time origins in order to obtain $C_\ell^{\mu\nu}(k,t)$

4. Take the time Fourier transform (again, using a trapezoid method) of $C_\ell^{\mu\nu}(k,t)$ in order to obtain the Fourier space longitudinal current-current correlation function, $C_\ell^{\mu\nu}(k,\omega)$

Note that in the preceding algorithms, the procedure is carried out for every combination of μ and ν.

In order to determine whether our implementation of the calculation of $S^{\mu\nu}(k,\omega)$ and $C_\ell^{\mu\nu}(k,\omega)$ was correct, we carried out several verification calculations. One such calculation was for the well-known Yukawa one-component plasma (YOCP). To do this we simply used the Yukawa potential as the ion-ion pair potential for several MD simulations. The results of our calculation of $S^{\mu\nu}(k,\omega)$ versus those of Mithen are shown in Fig. 35. Note that our simulations were much smaller and shorter than those of Mithen, as necessitated by the time frame of the workshop and the fact that this was only a verification of our calculation. In order to verify the calculation of $C_\ell^{\mu\nu}(k,\omega)$, we used it to indirectly calculate $S^{\mu\nu}(k,\omega)$ which showed good comparison to Mithen's results and to our direct calculation of $S^{\mu\nu}(k,\omega)$ [44]. The DSF plots are shown in Fig. 35.

Two observations one may glean from the comparisons of the DSF to Mithen's results are as follows: first, the direct calculation of the DSF is significantly noisier than the calculation via the LCCF (via division by ω^2); this effect is especially visible here due to the small simulation size but fades as the number of particles and time steps is increased. Moreover, the LCCF calculation, though typically a smoother result than the direct calculation, often suffers from spurious features near $\omega = 0$ due to the division by ω^2, which exaggerates any low-ω noise. We found, however, that these spurious features are not significantly abated as we increase the MD simulations size. For this reason, the direct calculation was used for our final DSF results; we simply increased the simulation size as needed to eliminate statistical noise.

In addition to comparison to Mithen's results, we also verified the direct calculation of the DSF by comparing it to our previously calculated static structure factor, $S(k)$, via the sum rule (17). An additional verification for $C_\ell^{\mu\nu}(k,\omega)$ was performed by comparing the time-domain initial value to the value predicted by theory (22).

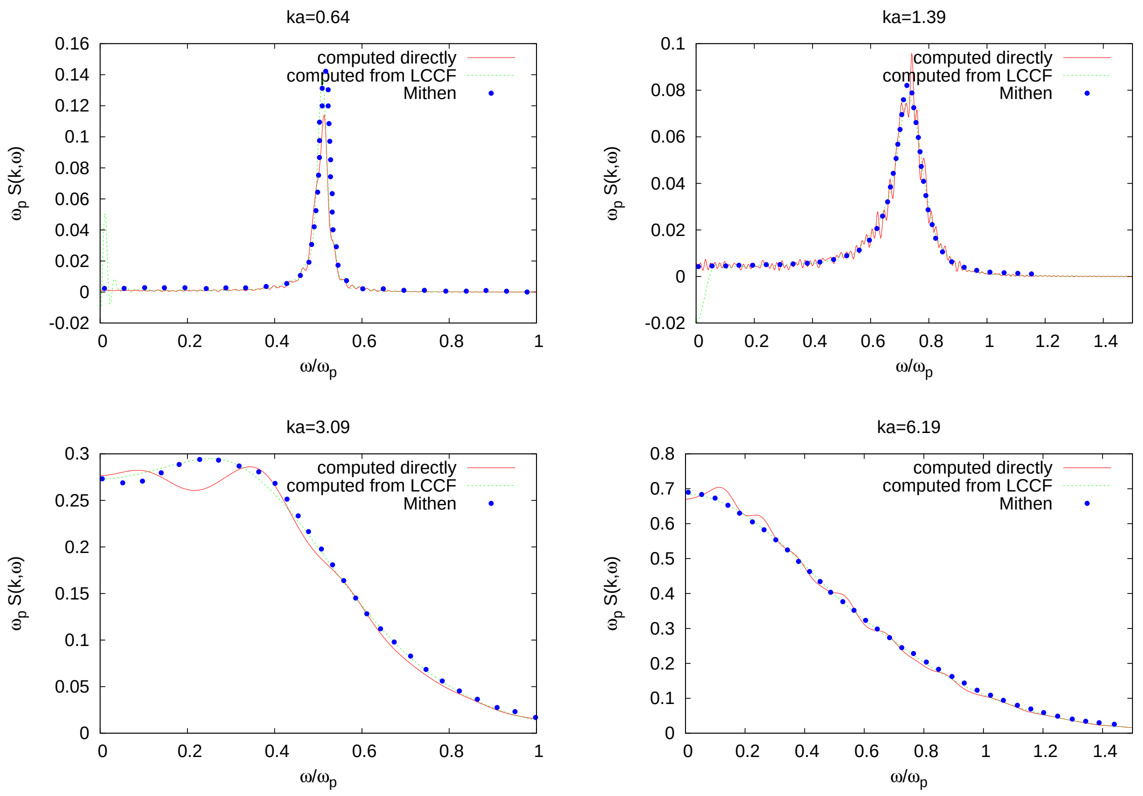


Figure 35: Plots of the dynamical structure factor as a function of ω for a Yukawa system, computed directly and from the LCCF, compared to the results of Mithen. Plots shown are for $\Gamma = 50$ (Γ is the "coupling parameter"—the ratio of electron kinetic energy to coupling) and $ka = 0.64$, 1.39, 3.09, and 6.19, where a is the average interatomic distance. The results of 10 MD simulations using 2500 particles and 10,000 post-equilibrium time steps ($\Delta t = 24$ as) were block-averaged to obtain these results.

Results for the static structure factor in aluminum and an iron-helium mixture

The static structure factor, $S(k)$, is of particular interest due to the fact that it is experimentally accessible via X-ray scattering experiments and can be used to determine the thermodynamic properties of a system. We used the results of MD simulations to calculate $S(k)$ for an aluminum system (Fig. 36 and a mixture of helium and iron ions (Fig. 37, 38, and 39). In order to verify those results, we compared our direct calculation of $S(k)$ to an approximate result calculated via use of the hypernetted-chain (HNC) equation and had very good agreement. The statistical noise visible in the direct calculation reduces as the size and length of the MD simulation are increased.

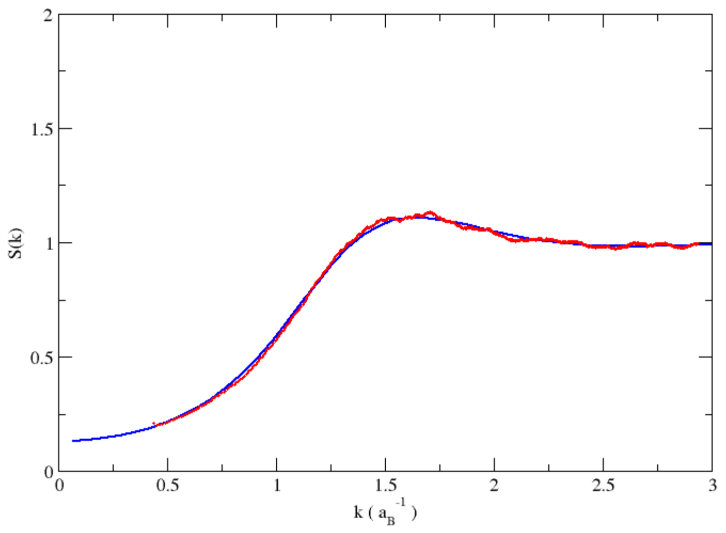


Figure 36: $S(k)$ for aluminum at a temperature of 5 eV and a density of 2.7 g/cm^3 as calculated directly from MD results using 2500 particles and 10,000 time steps ($\Delta t = 24$ as). Our results in red, results from the hyper-netted chain (HNC) approximation in blue.

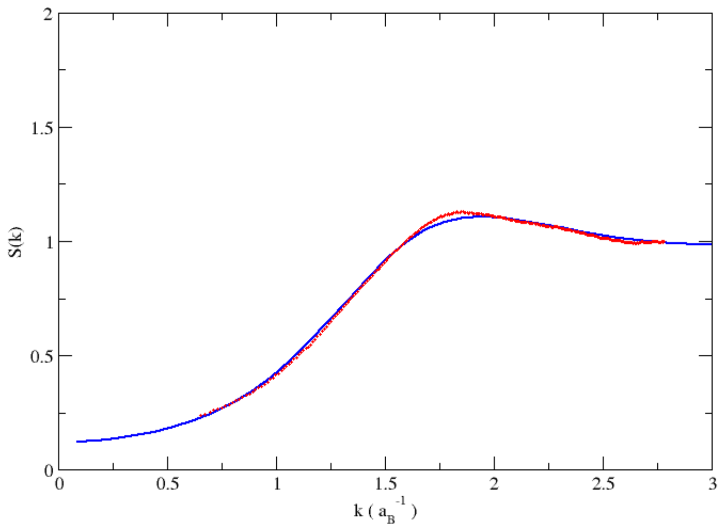


Figure 37: The iron-iron partial SSF for an iron/helium mixture at a temperature of 50 eV and a density of 10 g/cm^3 as calculated directly from MD results using 2500 particles and 10,000 time steps ($\Delta t = 24$ as). Our results in red, HNC results in blue.

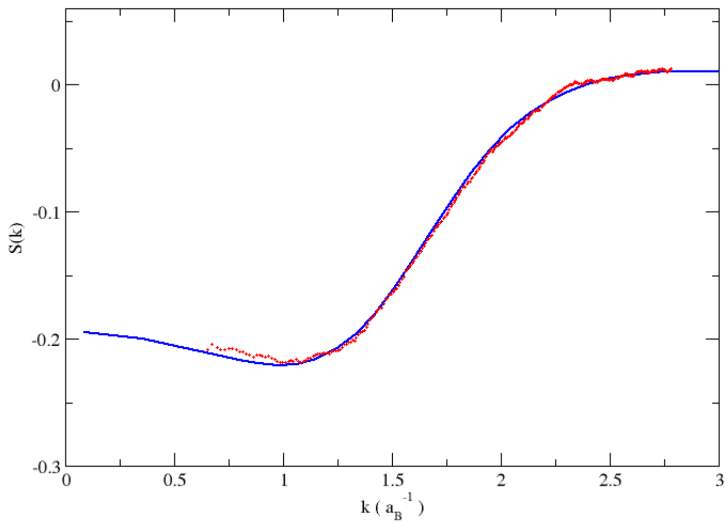


Figure 38: Iron-helium partial SSF for the same iron/helium mixture, compared to HNC result. This was computed by averaging both off-diagonal SSFs (which guarantees a real result).

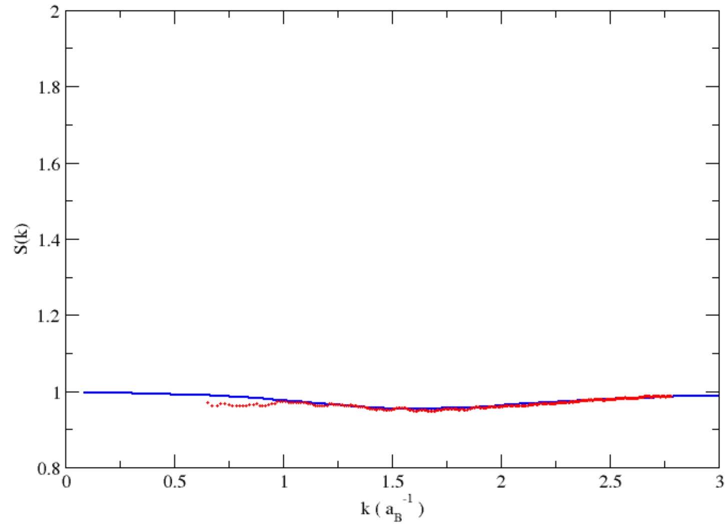


Figure 39: Helium-helium partial SSF for the same iron/helium mixture, compared to HNC result.

We also calculated $S(k)$ for a binary ionic mixture of iron and helium atoms at 50 eV and 10 g/cm^3, the results of which are shown in Figs. 37 to 39.

Results for the DSF in warm dense aluminum

Recent papers by Rüter and Redmer [59] and by White et al. [71] have produced results for the DSF in warm dense aluminum using, respectively, Kohn-Sham density-functional theory (KS DFT) and orbital-free density-functional theory (OF DFT). Here, we present our corresponding results for warm dense aluminum. Our results were all obtained from block-averaging the results of 10 MD simulations using 5000 particles and 40000 post-equilibrium time steps with a Δt of 24 as.

First, we compare our results to KS DFT (Fig. 40), which is thought to be accurate but suffers from high computational expense. We see good agreement between our results and the KS DFT results especially at larger k. Model differences become more salient at lower k because in this regime the electrons behave collectively and correlations are more important. Besides differences in the models, the differences between the two sets of results could be partially due to finite size effects, since KS DFT simulations are limited to small particle numbers. It should also be noted that exact agreement in k was impossible due to our periodic MD boundary conditions.

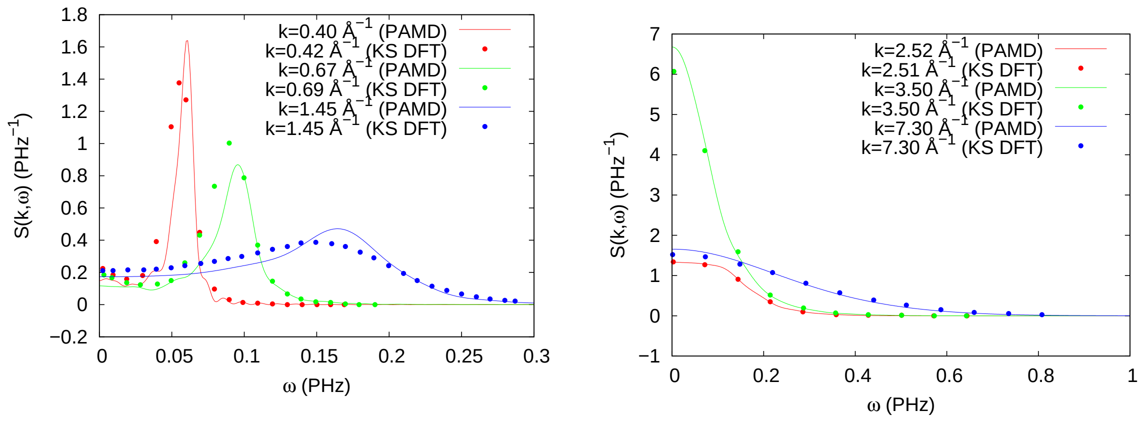


Figure 40: Plots of the DSF using PAMD (solid lines) and using KS DFT (points) in warm dense aluminum at 3.5 eV and 5.2 g/cm^3, for $k = 0.42$, 0.69, 1.45, 2.51, 3.50, and 7.30 Å^{-1}.

In Fig. 41, we compare our results to that from the OF DFT calculations of White et al. The agreement here is worse, but this is to be expected: White et al. used a pseudopotential designed to recover the bulk properties of aluminum, whereas PAMD uses instead an all-electron calculation. It is well known that OF DFT calculations do not recover bulk properties, since they ignore important quantum effects.

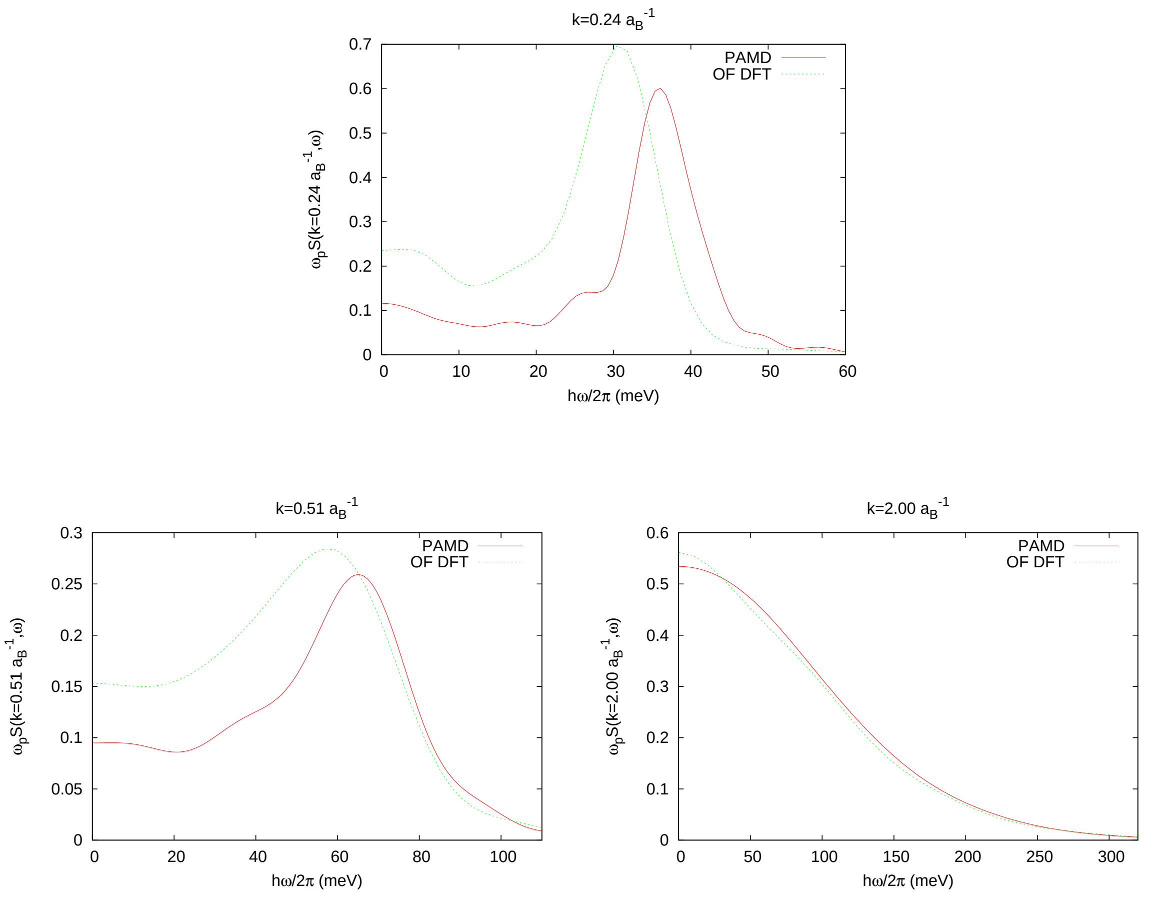


Figure 41: Plots of the DSF using PAMD (red) and using OF DFT (green) in warm dense aluminum at 5 eV and 2.7 g/cm^3, for $k = 0.24$, 0.51, and 2.00 a_B^{-1}

Fano-like Resonances in Mixtures

A Fano resonance is a feature in the line shapes of scattering cross-sections that is qualitatively observed by the presence of an asymmetric line shape. This phenomenon is ubiquitous in wave interactions and arises due to the interference between two interacting modes in the system, one mode being due to a background process and the other a resonant process. This type of resonance was first described as arising from interference between quantum states, but it has also been shown to occur in classical systems [33]. In the classical analogy, we consider two damped, coupled classical oscillators with some external driving force applied to one of the oscillators. In this system, the background process comes from the driving force and resonant mode is defined by one of the oscillatory modes of the coupled oscillators.

The asymmetry (and resulting anti-resonance) comes about when the driven oscillator experiences an equal force from the driving mechanism and the second oscillator, resulting in no net external force. This occurs in the real part of the oscillatory motion, but must also appear in the imaginary (dissipative) part [33, 63]. For this reason, one would expect this same Fano-like resonance to occur in the spectral analysis of strongly coupled systems.

One such strongly coupled system would be a binary ionic mixture in a warm, dense system. Since Fano resonances should be observable in the dissipative response of the system, we should expect to observe them in the DSF for the system. A forthcoming paper by Silvestri et. al discusses their prediction of the presence of Fano-like resonances in a binary ionic mixture using an idealized model for the matter and a description of the system based on the Quasi-Localized Charge Approximation (QLCA) [63]. Their prediction is shown in Fig.42.

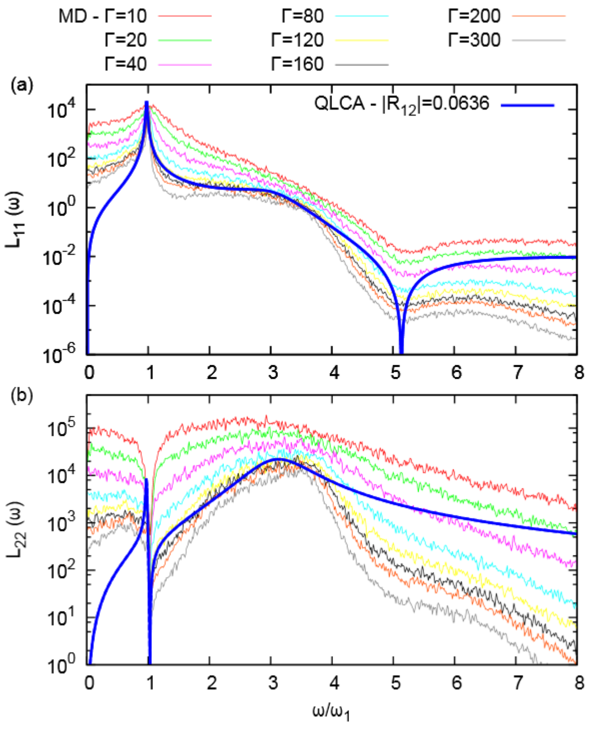


Figure 42: Current-current correlation function frequency spectrum. L_{11} and L_{22} refer to correlation function for species 1 and 2 in Silvestri et. al [63], respectively. Note the existence of the asymmetric line shape and deep anti-resonance in L_{22} for their QLCA results.

In order to see if the same asymmetric line shape indicative of a Fano resonance occurs in a real system, we ran MD simulations using the PAMD approach for an ionic mixture of helium and iron atoms. From the results we calculated $S(k,\omega)$ and $C_{\ell}(k,\omega)$ for this mixture. These calculations were performed near the end of the workshop period, as such the results are still preliminary but indicate the need for further investigation.

In Fig.43, one can see the asymmetric spectral profile indicative of a Fano resonance in the spectrum for the helium-helium correlation function (corresponding to S_{22} in the figure). The associated anti-resonance occurs at the same frequency where we see the resonant peak occur in the iron-iron correlation function (corresponding to S_{11} in the figure). This is consistent with Silvestri et. al's model predictions. The anti-resonance occurring in the lighter ion's response at the same frequency a resonance occurs in the heavier ion's response is also supported by the classical damped, harmonic oscillator analogy [33]. In this system, the perturbing force on the system (from the scattering radiation) is on resonance with the natural resonant frequency of the iron ions in the system while the motion of the helium ions in the system experience a damping effect, resulting in an anti-resonance.

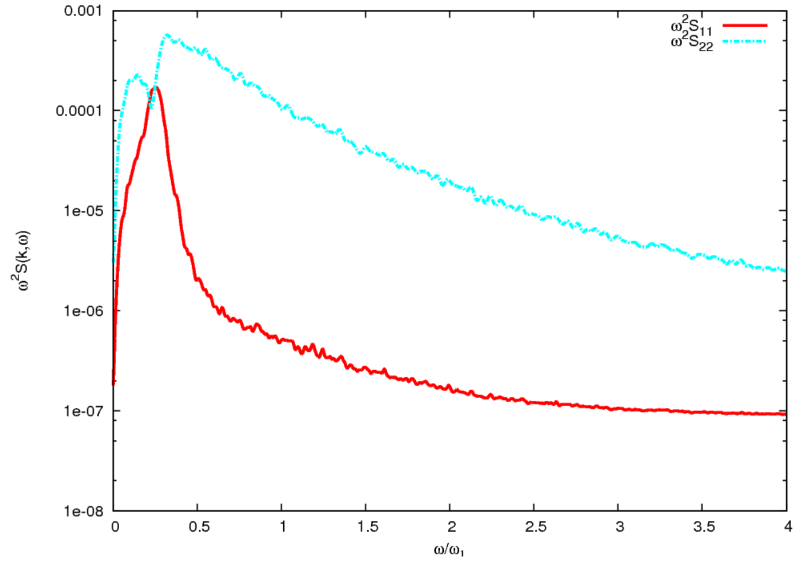


Figure 43: Current-current correlation function frequency spectrum. S_{11} corresponds to the iron-iron correlation function and S_{22} corresponds to the helium-helium correlation function. This calculation was done using an equal parts mixture of 2000 helium ions and 2000 iron ions at a temperature of 50 eV, density of 10 g/cm^3, for 1×10^5 time steps with a $\Delta t = 24$ as for $k = 0.1736 a_B^{-1}$

This being the first time that a calculation of this nature has been carried out for a mixture of realistic ions at warm dense matter conditions, these results hardly characterize the effects of interest. Further study is required, including determining the effect of mixing ratio, mass ratio (determined through choice of the species of the ions), density, and temperature on the behavior of this asymmetric line profile. The relative depth and position of the anti-resonance for different wavenumbers will also be studied so that this new phenomenon in warm dense mixtures can be more fully understood.

Conclusion

Our work during the 2014 Computational Physics Student Summer Workshop was to write codes to calculate the static structure factor and the ion-ion component of the dynamic structure factor and longitudinal current-current correlation function in warm dense matter systems. These calculations, which until now bordered on being computationally intractable, were computed quickly thanks to the information generated by classical MD simulations using the PAMD produced ion-ion pair potentials. With the PAMD approach developed by Charlie Starrett and Didier Saumon, the MD simulations could be run for long periods of time for a large system of ions, including binary mixtures. Our calculation was verified in several ways, including direct comparison to the well-known Yukawa one-component plasma results. Calculations for non-idealized systems were carried out, including warm dense aluminum and a binary ionic mixture of helium and iron ions. The results of the aluminum calculations show good agreement with well accepted results (from DFT-MD), but our calculations took far less time to carry out and represented a system of many more ions simulated over a longer time range. This indicates that the PAMD approach generates data representative of real physical systems and allows for the study of these systems in a computationally efficient way. The preliminary results from the calculations of a He-Fe mixture show features indicative of a Fano resonance, with a pronounced asymmetry and anti-resonance in the helium-helium LCCF at the resonant frequency seen in the iron-iron LCCF. The existence of this feature in the spectral profile of binary ionic mixtures is a new phenomenon and merits further study.

Our future work involves further characterizing these warm dense systems by determining the effect of density, temperature, and ion species type on the dynamic functions of interest. We also intend to further characterize the effect of changing simulation size, length, and time-step on the results of these calculations. The dispersion relation will be calculated by determining the location of resonant peaks in the DSF at various wavenumbers, allowing us to determine the speed of sound in the system. Now that the first experimental measurements of the ion-ion component of the DSF are available, comparison can be made with our calculations. In addition to the previous characterizations in single species systems, the effect of mass ratio (species type), mixing ratio, and wavenumber on the location and depth of the apparent anti-resonance will be studied in mixtures. Further investigation into what implications the existence of these Fano resonances might have will also be carried out.

Reducing Statistical Noise in Implicit Monte Carlo Thermal Radiation Transport

By A. Miguel Holgado and R. Tyler Holladay,
Todd J. Urbatsch, Allan B. Wollaber, and Mathew A. Cleveland (mentors)

Abstract

In this work, we apply noise reduction techniques to the Implicit Monte Carlo method for thermal radiation transport problems, which include source biasing and kernel density estimators (KDEs). Source biasing is applied in Milagro, Los Alamos National Lab's IMC code. Another IMC code, called DESERT, was written and determines quantities of interest such as material temperature using both tallies and KDEs. The KDE method was tested with an infinite medium problem with the material and radiation in thermal equilibrium. The material temperature estimated using KDEs was much smoother than that of the tally-estimated material temperature.

Introduction

This work focuses on the reduction of statistical noise that can manifest in the Fleck and Cummings Implicit Monte Carlo (IMC) method for simulations of thermal radiation transport (TRT) [21]. The IMC method is prone to violate the maximum principle if the spatial refinement is too fine or if the time step is too large. This is because energy deposition does not scale with the heat capacity as cells are refined. Milagro, an IMC TRT code developed at Los Alamos National Laboratory (LANL), was used to help characterize the problem. Basic source-biasing configurations were applied in Milagro and were shown to be insufficient to fully prevent maximum principle violation. For more complex noise reduction methods, a 2-D Monte Carlo code, called DESERT, was written and verified for simulating TRT using the IMC method. Verification problems include both particle transport with and without material feedback. In DESERT, kernel density estimator (KDE) methods were used to determine variables of interest and were compared to those estimated by histograms. KDEs have been previously been applied to Monte Carlo neutron transport problems for obtaining solutions with reduced variance [3]. We extend this application to TRT problems with the additional goal of preventing maximum principle violation in IMC simulations. Sample quantities are denoted with a hat, $\hat{f}$, and quantities estimated by tallies and KDEs are denoted with a dot, $\dot{f}$, and double-dots, $\ddot{f}$, respectively.

Thermal Radiation Transport

We consider the transport of photons in a purely-absorbing, homogeneous, static medium described by the gray TRT equations given by

$$\frac{1}{c}\frac{\partial I}{\partial t} + \vec{\Omega}\cdot\nabla I + \sigma_{\mathrm{a}} I = ac\frac{\sigma_{\mathrm{a}}}{4\pi}T^4 + \frac{Q}{4\pi} \, , \tag{25a}$$

$$c_{\mathrm{v}}\frac{\partial T}{\partial t} = \int_{4\pi} \sigma_{\mathrm{a}} \left(I - \frac{ac}{4\pi}T^4 \right) \, d\Omega \, , \tag{25b}$$

where I is the radiation intensity, Q is the radiation source, σ_{a} is the absorption opacity, c is the speed of light, a is the radiation constant, c_{v} is the material specific heat capacity, and T is the material temperature. Equation (25a) is the radiation transport equation and is coupled to equation (25b), the material energy balance equation. The radiation transport equation states that the temporal changes in the radiation field are due to the streaming of radiation out of the spatial volume, absorption of the radiation by the material, material emission of radiation, and sources. The material energy balance equation states that the temporal changes in the material temperature are due to the absorption and emission of radiation. Additionally, we assume that the radiation and material are in local thermodynamic equilibrium (LTE) such that the material emits radiation as a black body, which can be described by the Planck function. We assume that the material opacity is frequency-independent and use the gray approximation, which results in the Planckian emission term being proportional to T^4. The material energy density, u_{m}, and the radiation energy density, u_{r}, are defined in the following manner for convenience:

$$\frac{\partial u_{\mathrm{m}}}{\partial T} = c_{\mathrm{v}}(T) \, , \tag{26a}$$

$$u_{\mathrm{r}} = aT^4 \, . \tag{26b}$$

Using these relations (26), an exact relationship between u_{m} and u_{r} can be defined as

$$\beta = \frac{\partial u_{\mathrm{r}}}{\partial u_{\mathrm{m}}} = \frac{du_{\mathrm{r}}}{dT}\frac{dT}{du_{\mathrm{m}}} = \frac{4aT^3}{c_{\mathrm{v}}(T)} \, . \tag{27}$$

The units used in this work are tabulated in Table 1.

Table 1: Fundamental units.

Quantity	**Unit**
Length	centimeter (cm)
Time	shake (sh)
Angle	steradian (str)
Energy	jerk (jk)
Temperature	kilo-electron-volt (keV)

Using these units, the speed of light, c, takes on the value of 300 cm/sh.

Monte Carlo Method Overview

The Monte Carlo method (MCM) is a method that utilizes random sampling in order to numerically evaluate equations that are difficult to solve analytically. Here, the MCM is used to simulate photon transport through an optical medium in order to solve the IMC TRT equations, shown in equations (30a) and (30b). As can be seen, the IMC TRT equations are a set of 2 nonlinear partial differential integral equations. Because of their complexity, they cannot be solved analytically in all but the simplest of cases. In reality, the number of photons present in a system can be so large that it may be infeasible to simulate the transport of each one. To circumvent this, the total energy of the system at a given time is calculated and an energy weight (higher than the average photon energy) is assigned to a smaller number of Monte Carlo particles such that the total energy is the same [72]. The energy weight calculation for a Monte Carlo particle is shown in equation (28)

$$E_{system} = w_0 N_T \tag{28}$$

where E_{system} is the systems total energy at time t, w_0 is the energy weight of the Monte Carlo particle, and N_T is the number of particles to be simulated. To simulate particle transport, pseudo-random numbers between 0 and 1 are generated and used to determine a particles initial position, direction, distance to collision, and whether or not it is absorbed or scattered upon collision. The history of each particle is tracked and tallies are kept for events of interest such as location of emission, absorption, or scattering. Once the particle has been absorbed or left the system, a new particle is then simulated in the same process. For example, the particles distance to collision is calculated using equation (29), a derivation of which can be found in [72].

$$x = -\frac{\ln(\xi)}{\sigma_a} \tag{29}$$

where ξ is a pseudo-random number between 0 and 1 and σ_a is the absorption opacity (probability of absorption per unit distance). Boundary, initial, source, and material conditions also influence where particles originate and leave the system, how often absorption or scattering occurs, and the materials change in properties such as opacity and temperature. The tallies obtained from tracking the histories of the particles are then used to calculate quanitities of interest such as radiation and material temperature.

Variance Reduction

Due to the random sampling for various parameters affecting photon transport, the amount of particle interactions will vary from cell to cell. This effect is minimized for large coarse cells, where particles from the entire sampling distribution are included. In smaller more refined cells, however, there are fewer particle interactions, leading to a larger variance of properties such as temperature among the small cells. One particular problem related to this is nonphysical overheating, which is caused by a violation of the maximum principle, to be discussed and illustrated in further detail later. The variance of the solution goes as $\sim \frac{1}{N}$, where N is the number of particles initially sourced into the simulation. An example of how a solution responds to more particles used is shown in Figure 44.

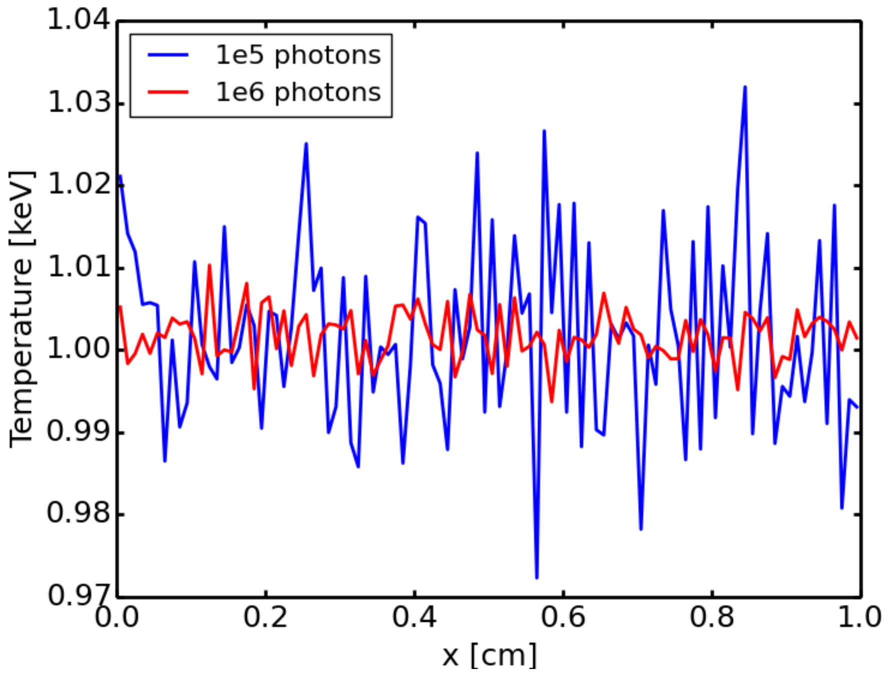


Figure 44: Variance of the material temperature for different initially sourced particles.

Even though the variance is indeed reduced with more particles, it can be become computationally expensive and prohibitive to reduce variance in this manner. Thus, it is necessary to apply variance reduction techniques to obtain more accurate solutions where very fine meshes are required. The methods we chose to reduce variance in this study include source biasing and kernel density estimation.

Implicit Monte Carlo Thermal Radiation Transport

Overview

The IMC method involves linearizing the Planckian emission term in the TRT equations and introduces effective-scattering, i.e. absorption followed immediately by re-emission. This can make the IMC solutions vulnerable to maximum principle violations where the material temperature can overheat past the boundary temperature set by the radiation. If the cell size is large enough and the time step is small enough, then the IMC solution will not violate the maximum principle [72]. The gray IMC equations are given by

$$\frac{1}{c}\frac{\partial I}{\partial t} + \vec{\Omega}\cdot\nabla I + \sigma_{\mathrm{a}} I = f\sigma_{\mathrm{a}} acT^4 + (1-f)\int_{4\pi} \sigma_{\mathrm{a}} I \, d\Omega \,, \tag{30a}$$

$$c_{\mathrm{v}}\frac{\partial T}{\partial t} = \int_{4\pi} f\sigma_{\mathrm{a}} I \, d\Omega - f\frac{ac\sigma_{\mathrm{a}}}{4\pi}T^4 \,, \tag{30b}$$

where f is the Fleck factor, given by

$$f = \frac{1}{1+\alpha\beta c\sigma_{\mathrm{a}}\Delta t} \, , \tag{31}$$

α is the implicitness of the scheme and Δt is the duration of the time step. The Fleck factor is the probability within a time step that an absorbed photon will not be re-emitted [72]. Using similar formulations from Wollaber [72], the material energy density is updated at the next time step, $n+1$, as

$$u_{\mathrm{m}}^{(n+1)} = u_{\mathrm{m}}^{(n)} - cf^{(n)}\sigma_{\mathrm{a}}u_{\mathrm{r}}^{(n)}\Delta t + \dot{E}_A \, , \tag{32}$$

where $\dot{E}_A$ is the tally-estimated average energy deposited from effective absorptions in a given cell and is given by

$$\dot{E}_A = \frac{1}{NV_{\mathrm{cell}}}\sum_{i=1}^{N} w_i \, . \tag{33}$$

The material temperature at the next time step can be calculated using

$$u_{\mathrm{m}}^{(n+1)} = \int_0^{T^{(n+1)}} c_{\mathrm{v}}(T)\, dT \, . \tag{34}$$

In this work, we consider c_{v} to be constant, so that the update for the material temperature becomes

$$T^{(n+1)} = \frac{u_{\mathrm{m}}^{(n+1)}}{c_{\mathrm{v}}} \, , \tag{35}$$

Maximum Principle Violation

The maximum principle is a limitation on the maximum amount the temperature can change for a given cell size and timestep. The smaller a cell is the closer it is to the threshold of maximum principle violation.

For illustration, we consider the Marshak wave problem where a hot isotropic wave of photons impinges on the face of an initially-cold material. The IMC method is used to characterize the problem and track the evolution of the material temperature, which is simulated in Milagro. We consider a 1-D slab geometry with a length of 10 cm and an initial material temperature of 10^{-6} keV. A wave of radiation impinges on the left side of the slab at a temperature of 1 keV. The first 9 cm of the slab are discretized into 9 equally spaced cells, while the last 1 cm is discretized into 1000 equally spaced cells. This is illustrated in Figure 45.

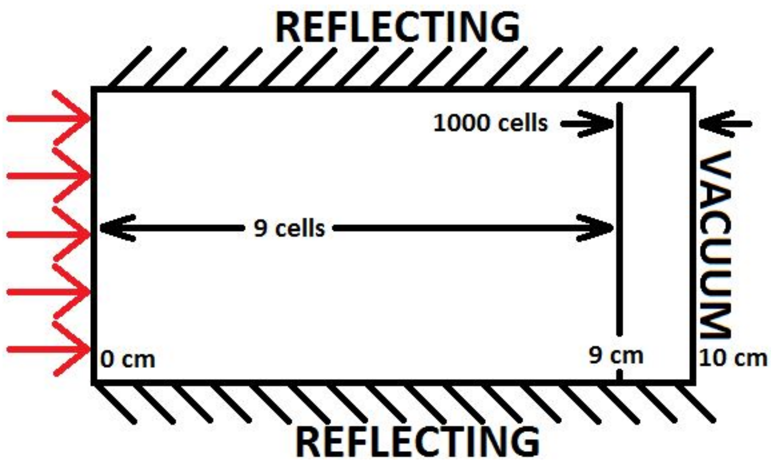


Figure 45: 1-D Marshak wave problem with a coarse region on the left and a very refined region on the right.

The problem is set to evolve for 10 sh at a time step of 0.01 sh and the material temperature is tracked as the radiation propagates through the slab. The material temperature at 9 sh is plotted for 10^3 and 10^5 particles in Figure 46.

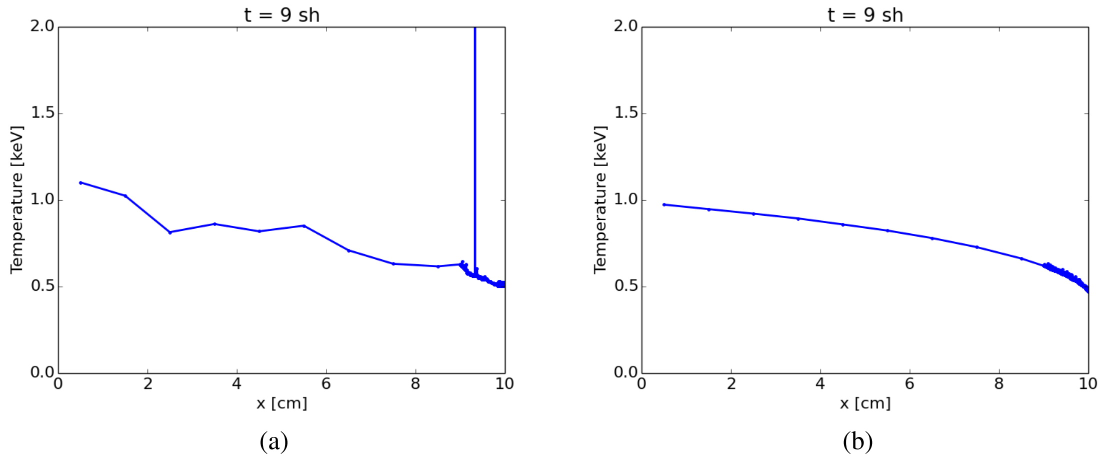
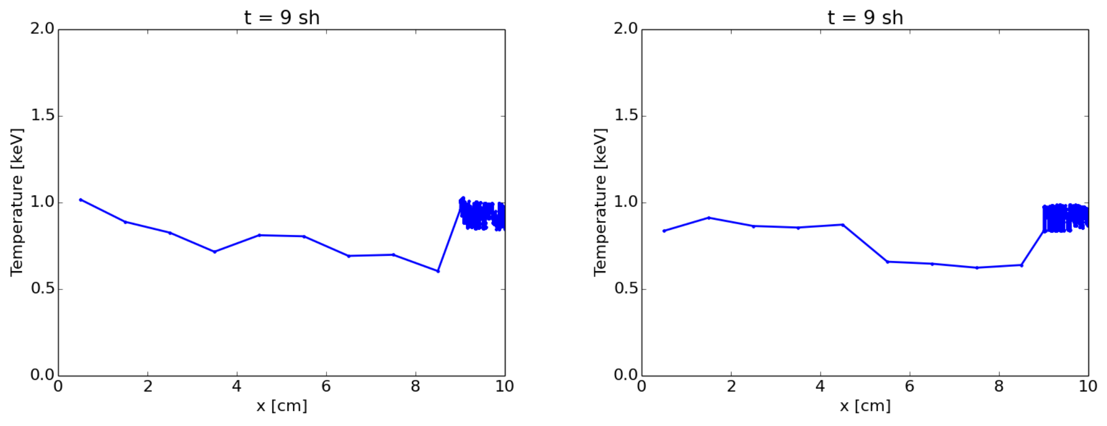


Figure 46: Material temperature at 9 sh for the Marshak wave problem. Left: 10^3 initially sourced particles. Right: 10^5 initially sourced particles.

The material temperature with 10^3 initially sourced particles experiences statistical noise in the coarse region and both overheating and statistical noise in the refined region. With 10^5 particles, the coarse region is relatively smooth and the refined region experiences some statistical noise, but no overheating. This indicates that increasing the number of particles for this problem setup can prevent nonphysical overheating. Thus we can conclude that with the given spatial and temporal parameters, the Marshak wave problem is on the cusp of maximum principle violation. The statistical noise that occurs during the simulation pushes the problem into the unstable region, causing overheating. If a relatively low amount of particles are sourced into the problem, the temperature in the highly refined region converges to an incorrect, nonphysical solution where the material temperature is higher than the impinging radiation temperature. This is illustrated in Figure 47, where 1 and 10 particles are used and the geometry and meshing are identical to that in Figure 46.



(a) (b)

Figure 47: Material temperature at 9 sh for the Marshak wave problem. Left: 1 initially sourced particles. Right: 10 initially sourced particles.

Not only has the temperature in the highly refined region converged to the wrong solution, but there is also significant statistical noise in that region. These results show that in order to get accurate solutions, the number of particles used must be great enough to reduce the statistical noise such that the maximum principle is not violated and that IMC method does not converge to the wrong solution.

Source Biasing

An exploration of source biasing was done in Milagro to determine its feasibility for eliminating nonphysical spikes and adequately reducing variance. Source biasing reduces the variance in a particular region by increasing the number of sampled particles in that region. To apply source biasing, each of the cells in a problem is assigned a bias relative to the other cells. Subsequently, via equations (36) and (37) the number of particles sampled in a particular cell is determined.

$$W_I = \frac{\sum S_i b_i}{N_T} \tag{36}$$

$$N_i = \frac{S_i b_i}{W_I} \tag{37}$$

W_I is the biased ideal particle energy weight, S_i is the source funtion of the i^{th} cell, b_i is the bias applied to the i^{th} cell, and N_T is the total number of particles to be used in the simulation. Equations (36) and (37) also represent how particles are sampled in an unbiased problem when each b_i is set to the same value (i.e. no one cell is biased more than any other).

3 different source biasing configurations have been applied to the 1-D problem illustrated above. This is to show the effect on variance reduction and to gain a better understanding of

how to better apply source biasing. The first configuration, shown in Figure 48 is a bias of 1000 applied to the coarse region on the left. The second configuration, shown in Figure 49 is a bias of 1000 applied to the refined region on the right, and finally Figure 50 is a bias ramped approximately linearly from left to right from 1 to 1000. The bias increases by approximately 111 per cm such that the left most centimeter has a bias of 1 and the right most has a bias of 1000.

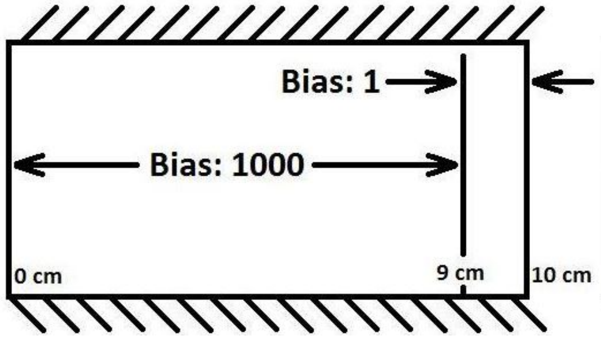


Figure 48: A bias of 1000 applied in the coarse region.

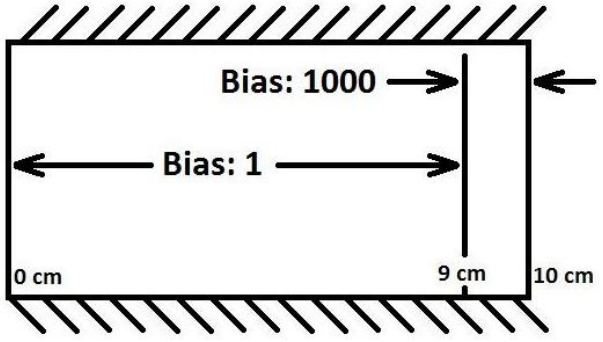


Figure 49: A bias of 1000 applied in the refined region.

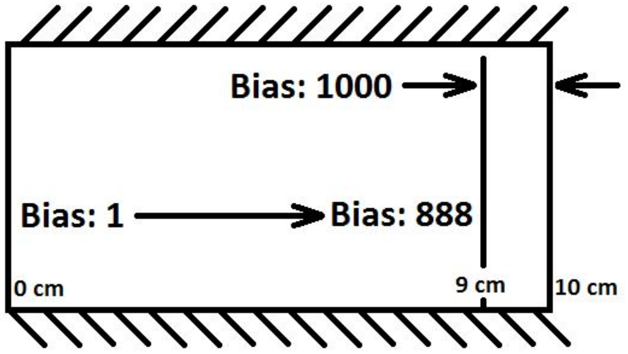


Figure 50: An approximate linearly ramped bias from 1 on the left to 1000 on the right.

With all other variables the same, simulations were run to determine if the refined or ramped biasing configurations above would be adequate for eliminating nonphysical over heating. The results of these 2 simulations along with results for simulations with no bias and bias in the coarse region are presented below.

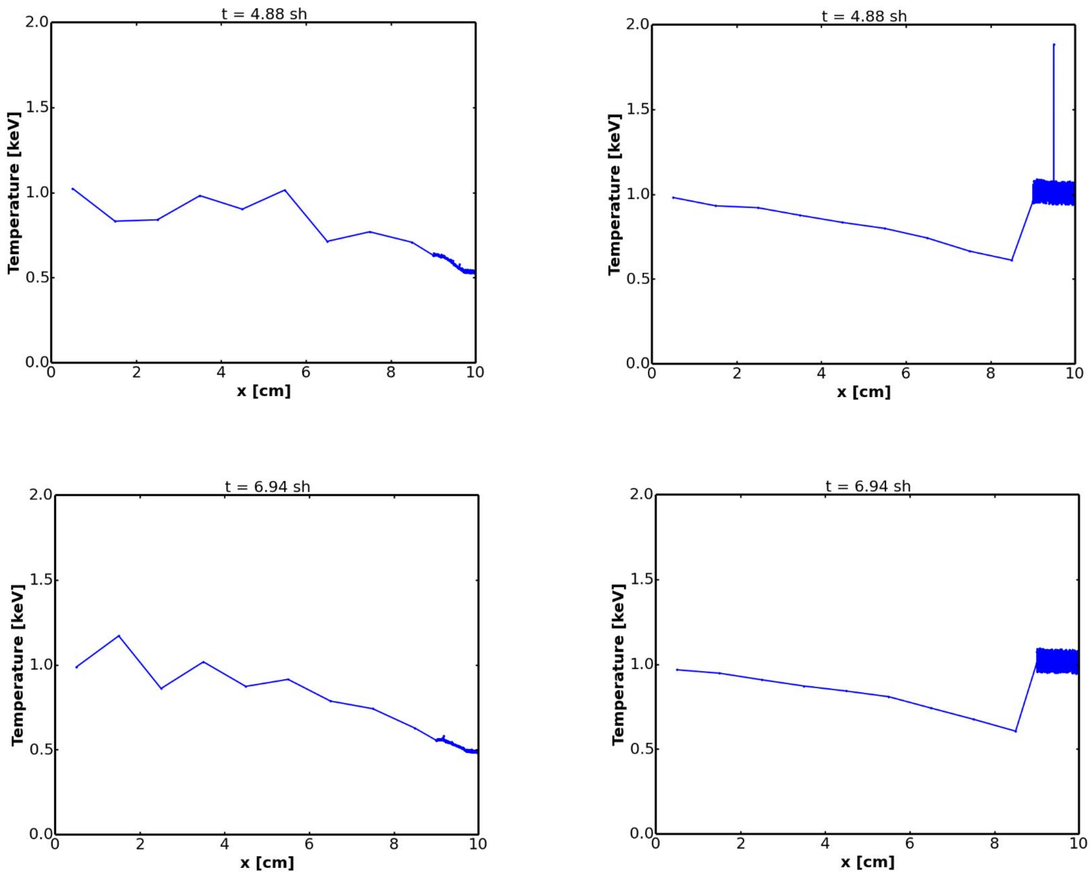


Figure 51: Material temperature at 4.88 and 6.94 sh. Left: No source biasing applied. Right: Bias applied in the coarse region on the left.

As can be seen in Figure 51 on the left, with no biasing present the solution in the coarse region fluctuates and the fine region converges to the expected solution. While there are no spikes present in the timesteps shown, they do occur frequently throughout the course of the simulation. Figure 51 on the right shows the results of bias being applied in the coarse region. While obviously the course region is much smoother, the refined region to converge to the wrong solution. This is because many more of the particles are now being sampled in the coarse region and thus fewer are being sampled in the refined region. Consequently, the variance in the refined region has increased such that it no longer converges to the correct solution.

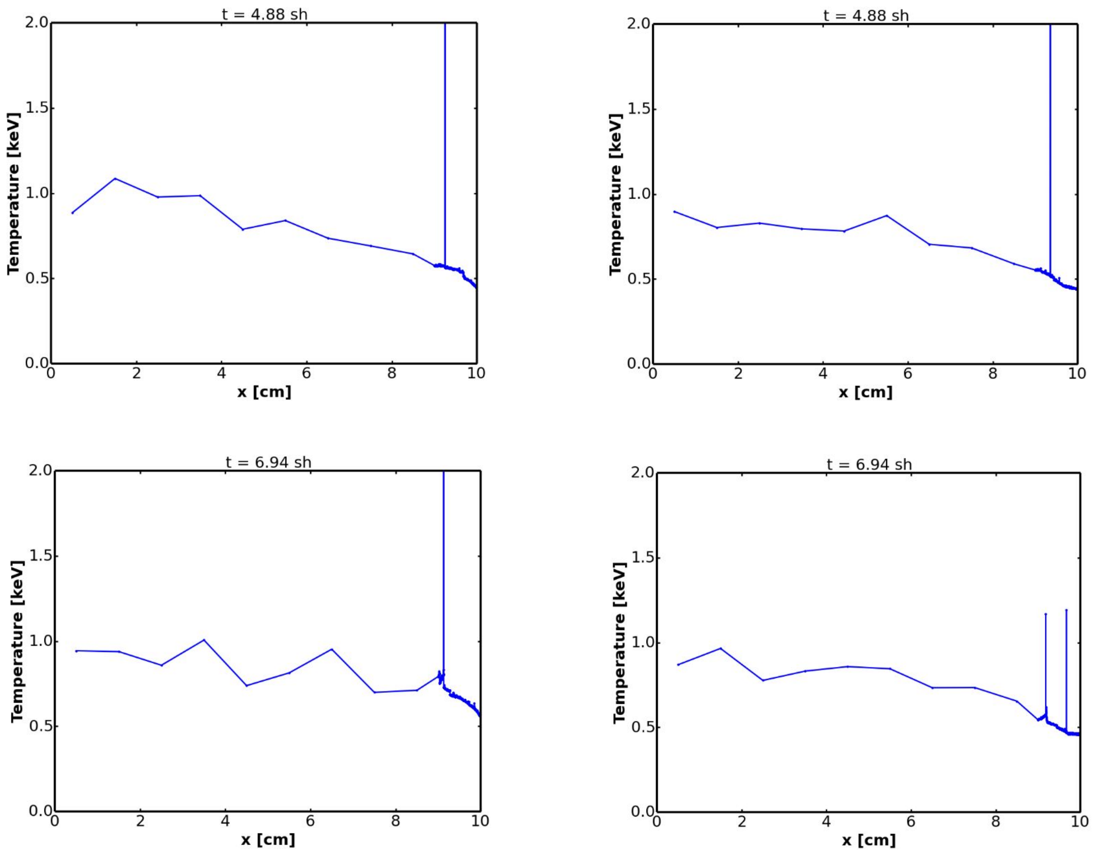


Figure 52: Material temperature at 4.88 and 6.94 sh. Left: Bias applied in the refined region on the right. Right: Bias ramped from 1 in the first centimeter to 1000 in the final centimeter.

Figure 52 on the left shows the results of biasing in the refined region. While this region now converges to the correct solution, the nonphysical spikes are still present and the coarse region now fluctuates greatly. It is important to note however that the spikes are observed to occur less frequently when the simulation is viewed in whole. Finally, the ramped biasing solution is shown in Figure 52 on the right. As can be seen, the solution becomes smoother from left to right however the nonphysical spikes are still present. Again it should be noted that the nonphysical spikes occur even less frequently when viewing the entire simulation than when the biasing is only done in the refined region.

From the results of these preliminary trials, it is obvious that further exploration needs to be done to determine if source biasing will be a feasible method in eliminating the nonphysical spikes. The ramped biasing configuration, while still inadequate, was the most successful in eliminating the spikes. Further exploration is beginning now to improve upon this configuration by introducing physics based source biasing. One method to be tested is basing the biasing on the mean free path of a particle in addition to the distance away from the refined region. This will further improve statistics in the refined region by eliminating the extra bias being applied in the regions where particles are very unlikely to reach the refined region (i.e. the coarse cells on the far left of the problem).

Additionally, the effects of these biasing configurations are also being explored in relatively optically thin problems. With an opacity of 1 cm^{-1} and a width of 10 cm, the problem above which is 10 mean free paths thick, is considered to be relatively optically thick. Simulations are currently being done on problems of the same opacity but with a width of 1 cm. Finally, we also plan to implement and test other biasing techniques such as directional biasing and weight windowing. With continued study of more in depth methods of source biasing, it will be possible to compare the overall efficiency of the these biasing techniques with other variance reduction methods. This will allow us to best implement variance reduction techniques for various types of problems and improve the efficiency of the IMC TRT simulations.

Kernel Density Estimation

Theory

A detailed introduction to statistical analysis via kernel density estimation can be found in [62]. We introduce the concept of kernel density estimation by first reviewing data representation using histograms. Histograms can represent a given probability density function (pdf) by "stacking" observed data points, $\hat{x}_i$, into bins. The estimated pdf via histogram can be expressed as

$$\hat{f}(x) = \frac{1}{Nb}\left(\#\text{ of } \hat{x}_i \text{ in same bin as } x\right) . \tag{38}$$

In order for a histogram to adequately resolve a pdf, a large number of bins and data samples are required. Kernel density estimation is a nonparametric density estimation technique that uses a kernel function to represent an observed data point, $\hat{x}_i$. Given a set of N independent and identically distributed sample data, $\{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_N\}$, from an unknown density function, $f(x)$, the kernel density estimated function is given by

$$\hat{f}(x) = \frac{1}{Nh}\sum_{i=1}^{N} k\left(\frac{x-\hat{x}_i}{h}\right) , \tag{39}$$

where k is a given kernel function and h is the bandwidth of the kernel. An example of how data representation is different between histograms and KDEs is given in Figure 53, where 6 samples are taken from an unknown pdf.

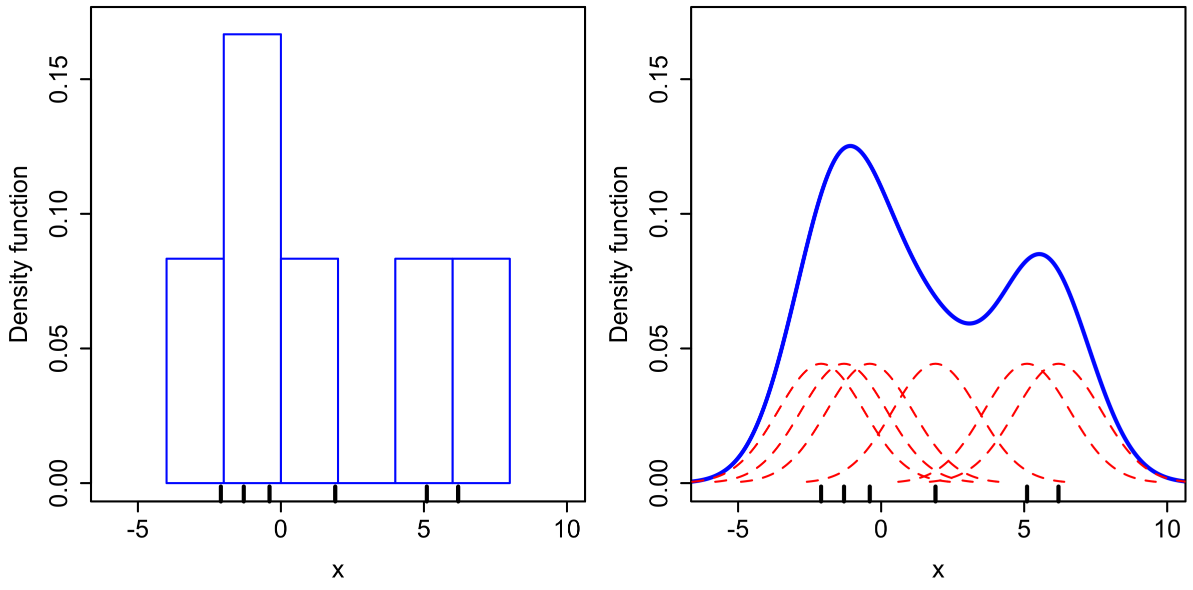


Figure 53: Data representation differences between histograms and KDEs.

In the histogram, the 6 samples are stacked into their respective bins, while in the KDE, each kernel is represented with a kernel function and all the individual kernels are summed to obtain the final estimate of the pdf. The smoothness of the histogram mainly depends on how many bins are used. An example is shown in Figure 54.

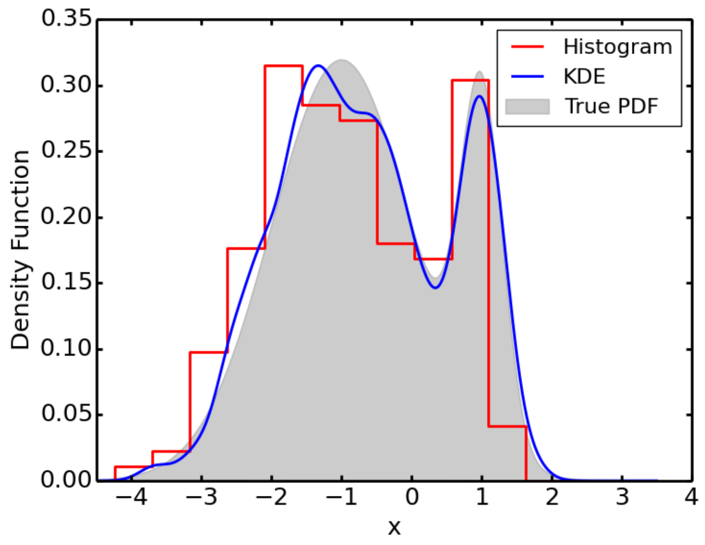


Figure 54: Smoothness of histograms and KDEs.

In the context of IMC TRT, the smoothness of the histogram depends on how many spatial cells are used; however, it was previously shown that higher spatial refinement can cause nonphysical overheating. We use KDEs to mitigate this problem while obtaining a smooth solution.

Bandwidth selection is important when using KDEs. In Figure 55, several samples are taken from a Gaussian distribution and KDEs are used to estimate the pdf from the given samples at various bandwidths.

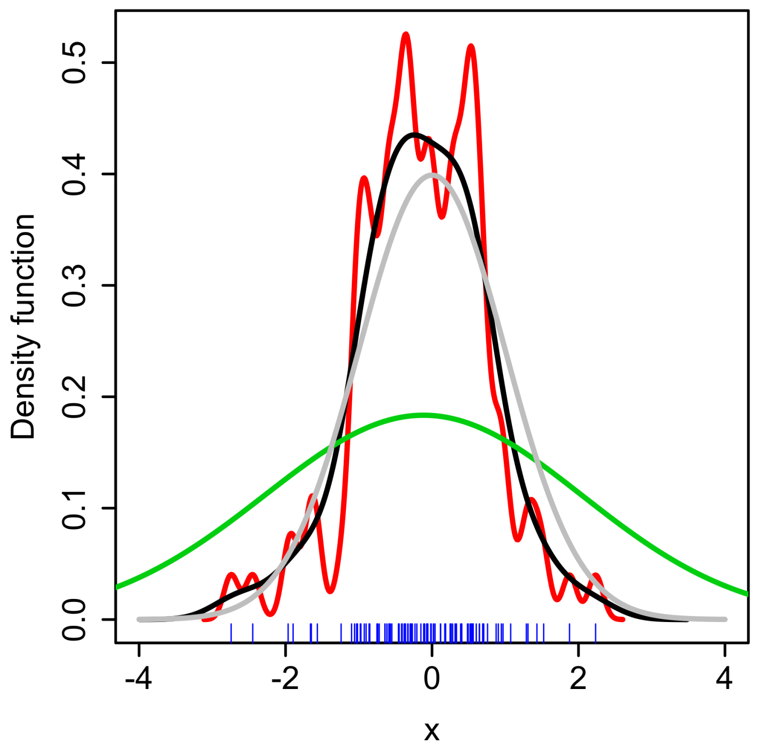


Figure 55: KDEs at various bandwidths for samples taken from the Gaussian distribution which is given by the gray curve. The red curve indicates the use of a small bandwidth. The green curve indicates the use of a large bandwidth. The black curve indicates the use of an optimally-sized bandwidth.

Using a KDE that is undersmoothed will result in an estimate that has sharp peaks for every sample that is observed, which does not convey information about the underlying features of the unknown pdf. Using a KDE that is oversmoothed will result in an estimate where all the information that the samples give is smeared throughout the domain. The rescaled Epanechnikov kernel is given by

$$k_e(x) = \begin{cases} \frac{3}{4\sqrt{5}}\left(1 - x^2/5\right), & |x| \leq \sqrt{5} \\ 0, & \text{otherwise} \end{cases} . \tag{40}$$

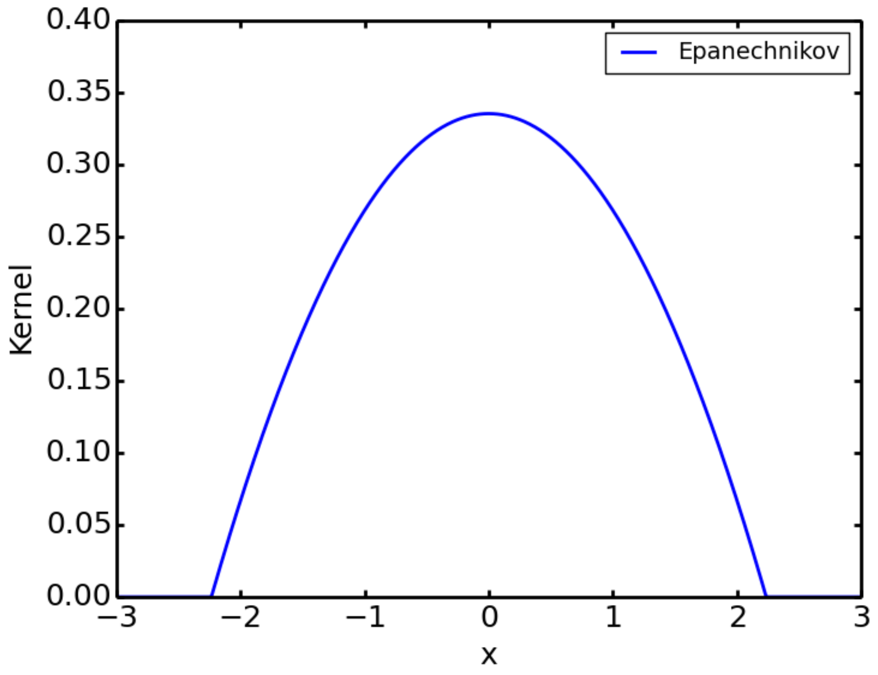


Figure 56: The Epanechnikov kernel.

where s is the standard deviation of the samples, and N is the number of samples []. The Epanechnikov kernel has compact support, so data observed within its bandwidth will only contribute to the values of the encompassed nodes. The optimal bandwidth for the Epanechnikov kernel is given by

$$h_{\text{opt}} = s\left(\frac{4}{5N}\right)^{1/7} . \tag{41}$$

KDE Boundary Correction

KDEs cannot resolve regions near the boundary well, so boundary correction methods are required in order for the KDE to approximate the pdf well enough over the entire problem domain. In this work, we only consider the reflective boundary correction method. This method maps the observed data samples beyond the boundary, which effectively creates mirror images of the data points. If the boundary is at the origin, then the boundary-corrected KDE is given by

$$k(x) = \frac{1}{Nh}\sum_{i=1}^{N}\left[k\left(\frac{x-\hat{x}_i}{h}\right)+k\left(\frac{x+\hat{x}_i}{h}\right)\right] . \tag{42}$$

If a boundary is located a distance L away from the origin, then the domain point, x, and sample point, $\hat{x}_i$, need to be translated a distance L. The boundary-corrected KDE can thus be expressed as

$$k(x) = \frac{1}{Nh}\sum_{i=1}^{N}\left[k\left(\frac{x-\hat{x}_i}{h}\right)+k\left(\frac{(x-L)+(\hat{x}_i-L)}{h}\right)\right] , \tag{43}$$

$$k(x) = \frac{1}{Nh}\sum_{i=1}^{N}\left[k\left(\frac{x-\hat{x}_i}{h}\right)+k\left(\frac{x+\hat{x}_i-2L}{h}\right)\right] . \tag{44}$$

Monte Carlo Tallies using KDEs

Monte Carlo TRT uses histogram tallies to estimate variables of interest. The scalar flux is estimated by the collision tally using

$$\hat{\phi}(x,y) = \frac{c_i}{NV_i\sigma_{\mathrm{t}}} , \tag{45}$$

where c_i is the number of collisions in the volume V_i, and N is the number of histories. The scalar flux is estimated by the KDE using

$$\hat{\phi}(x,y) = \frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C}\frac{w_{i,c}}{\sigma_{\mathrm{t}}}\frac{1}{h_x}k\left(\frac{x-\hat{x}_{i,c}}{h_x}\right)\frac{1}{h_y}k\left(\frac{y-\hat{y}_{i,c}}{h_x}\right) , \tag{46}$$

whose variance is given by

$$s_{KDE}^2 = \frac{1}{N-1}\left(\frac{1}{N}\sum_{i=1}^{N}\left[\sum_{c=1}^{C_i}\frac{w_{i,c}}{\sigma_{\mathrm{t}}}\frac{1}{h_x}k\left(\frac{x-\hat{x}_{i,c}}{h_x}\right)\frac{1}{h_y}k\left(\frac{y-\hat{y}_{i,c}}{h_y}\right)\right]^2-\phi_{KDE}^2\right) . \tag{47}$$

We now apply the KDE to the material variables. The KDE form of the mean energy deposited from effective absorptions, $\dot{E}_A$, is derived in the same manner as how the KDE form of the scalar flux was derived for neutron transport in Banerjee [3]. The normalized distribution of energy deposited in the problem domain from effective absorptions is given by

$$\frac{E_A(x)}{\int E_A(x)\,dx} = \frac{1}{\sum_{i=1}^{N}w_i}\frac{1}{NV_i}\sum_{i=1}^{N}\frac{w_i}{h_x}k\left(\frac{x-\hat{x}_i}{h_x}\right) , \tag{48}$$

where

$$\int E_A(x)\,dx = E_AV = \frac{1}{N}\sum_{i=1}^{N}w_i , \tag{49}$$

so that the KDE form of the mean energy deposited in a cell due to effective absorption is given by

$$\ddot{E}_A(x) = \frac{1}{N}\sum_{i=1}^{N}\frac{w_i}{h_x}k\left(\frac{x-\hat{x}_i}{h_x}\right) , \tag{50}$$

with the 2-D equivalent given by

$$\ddot{E}_A(x,y) = \frac{1}{N}\sum_{i=1}^{N}w_i\frac{1}{h_x}k\left(\frac{x-\hat{x}_i}{h_x}\right)\frac{1}{h_y}k\left(\frac{y-\hat{y}_i}{h_y}\right) , \tag{51}$$

The KDE form of the material-energy-update equation is thus given by

$$u_{\mathrm{m}}^{(n+1)} = u_{\mathrm{m}}^{(n)} = cf\sigma_{\mathrm{a}}u_{\mathrm{r}}^{(n)}\Delta t+\frac{1}{N}\sum_{i=1}^{N}w_i\frac{1}{h_x}k\left(\frac{x-\hat{x}_i}{h_x}\right)\frac{1}{h_y}k\left(\frac{y-\hat{y}_i}{h_y}\right) , \tag{52}$$

and the material temperature is updated using (35).

DESERT: Code Design

A 2-D parallel code was written in C++ to simulate time-dependent gray-TRT using IMC in Cartesian geometry. The code's name is DESERT, which stands for DEnSity-Estimated Radiation Transport. A detailed description of how to implement the IMC method for TRT is given in the Wollaber dissertation [72]. The angular dimension is represented by two angles: the polar angle, θ, and the azimuthal angle, ϕ. The direction cosines are given by

$$\mu = \cos\theta \ , \tag{53a}$$

$$\eta = \sin\theta \sin\phi \ , \tag{53b}$$

$$\xi = \sin\theta \cos\phi \ , \tag{53c}$$

Since we only track particles in 2-D, ξ, is not used in the code.

Table 2: Files that make up the Monte Carlo code. The $\vee$ symbol indicates a .cpp file. The † symbol indicates a .h file. The ‡ symbol indicates both a .h and .cpp file.

File Name	**File Description**
Constants †	Contains physical constants
Filter ‡	Contains filter functions
KDE ‡	Contains KDE functions
Materials ‡	Contains material properties and functions to update temperature
Mesh ‡	Constructs mesh
Output ‡	Formats output
Photons ‡	Contains phase-space variables of photons
RNG †	Generates random numbers
run $\vee$	Parses input, runs the code, and writes to output
StreamPhotons ‡	Transports particles in the problem domain
Tallies ‡	Tallies particle collision events

Energy conservation is tracked at each time step of the simulation. The energy balance at the end of each time step is

$$E_\mathrm{T} = E_\mathrm{D} + E_\mathrm{C} \ , \tag{54}$$

where E_T is the total amount of energy injected at the beginning of the time step, E_D is the total amount of energy deposited due to effective absorptions, and E_C is the total amount of energy in census for the next time step. Since numerical precision is finite, a balance measure, b, can be defined as

$$b = \frac{E_\mathrm{T} - (E_\mathrm{D} + E_\mathrm{C})}{E_\mathrm{T}} \ , \tag{55}$$

where a tolerance, ε, can be defined to be on the order of round-off error such that energy conservation is achieved if $b < \varepsilon$.

DESERT: Code Verification

No Material Feedback

We verify the case of a constant, isotropic, homogeneous source in steady state with no material feedback in an infinite, purely-absorbing medium described by

$$\mu \frac{\partial I}{\partial x} + \sigma_{\mathrm{a}} I = \frac{Q}{2} \ , \ \text{for } 0 \le x \le L \ , \tag{56a}$$

with reflective boundary conditions at all endpoints

$$I'(0) = 0 \ , \ I'(L) = 0 \ , \tag{56b}$$

The scalar flux solution to (56) is simply

$$\phi(x) = \frac{Q}{\sigma_{\mathrm{a}}} \ . \tag{57}$$

We consider a 1-D slab with a length of 10 cm along the x and y dimensions and reflecting boundary conditions along all sides. On this geometry, we compare the scalar flux solution of the 1-D collision flux tally, KDE, and analytic in Figure 57.

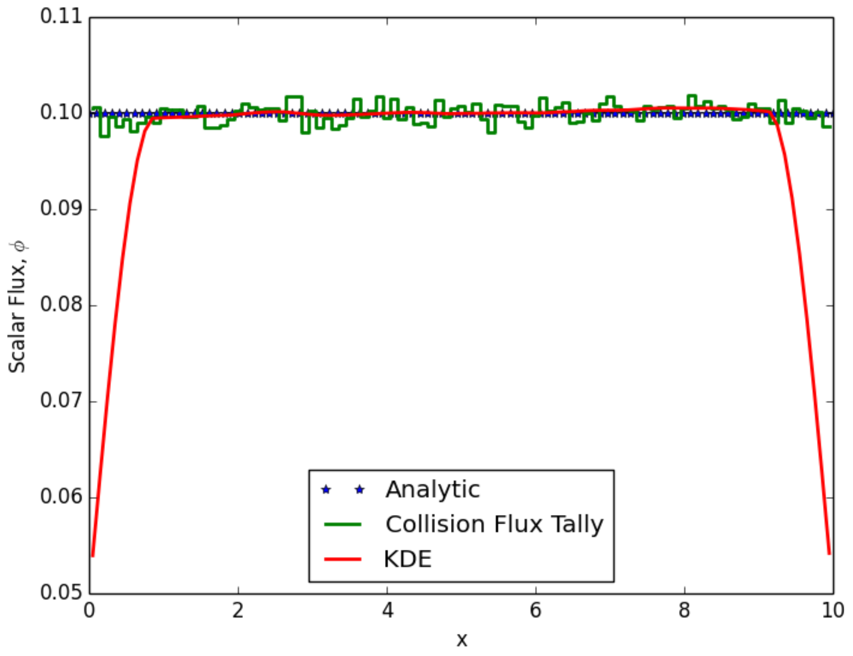


Figure 57: Scalar flux in an infinite medium with a homogeneous source estimated using tallies and KDEs. 100 cells and 10^6 particles are used.

The tally and KDE solutions agree with the analytic solution. While the tally solution is noisy, the KDE solution accurately approximates the scalar flux in the interior, but poorly approximates when approaching the boundary. We also note that the KDE solution is smooth, which is highly desirable. We apply a reflective boundary correction to the KDE at the origin and plot the results in Figure 58.

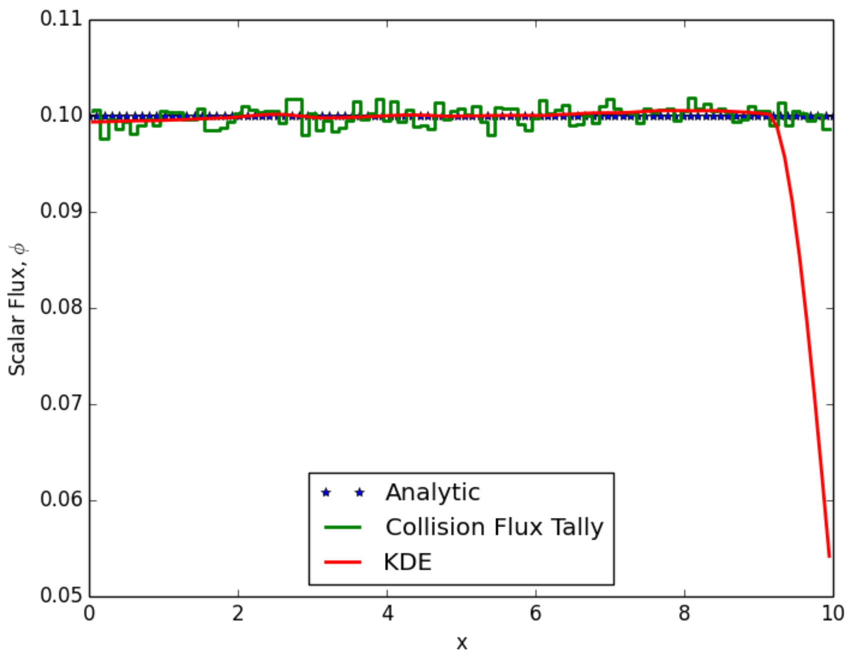
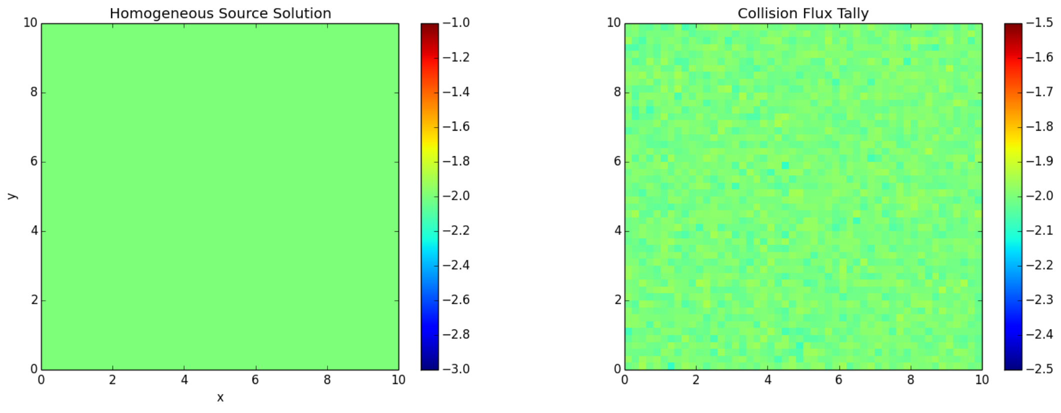


Figure 58: Same conditions as Figure 57 with boundary correction used at the origin.

With the boundary correction at the origin, the KDE solution when approaching the origin now agrees well with the analytic solution. We now consider 2-D geometry and plot the 2-D analytic solution and collision flux tally in Figure 59.



(a) Analytic

(b) Tally

Figure 59: Scalar flux in an infinite homogeneous medium in 2-D. 2500 cells and 10^6 particles are used.

The tally and analytic solution appear to agree well. In order to better characterize the noise in the tally solution, we define a residual given by

$$R = \left| f - \hat{f} \right| \,, \tag{58}$$

where f is the analytic solution and $\hat{f}$ is the Monte Carlo approximation. The residual for the 2-D infinite, homogeneous source case is plotted in Figure 60 on a logarithmic scale.

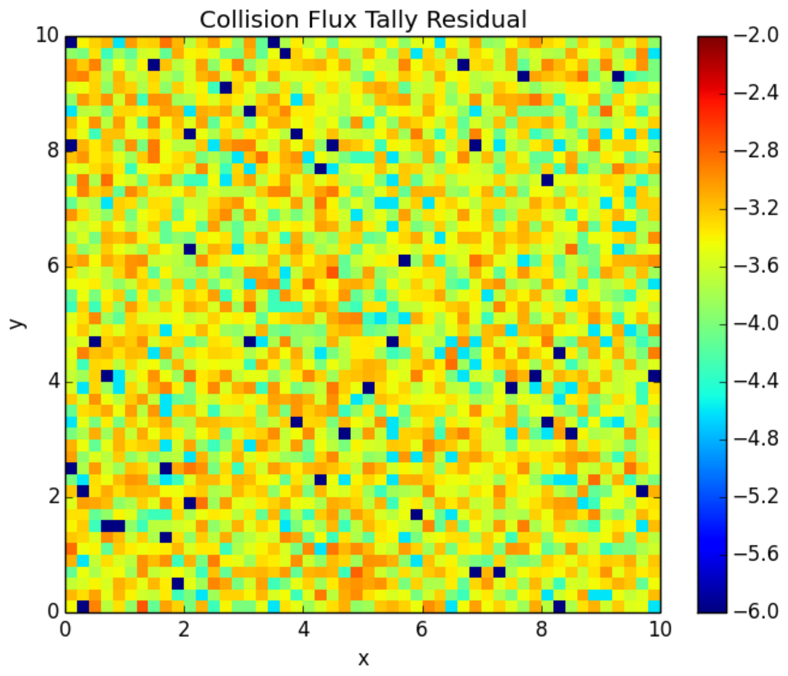
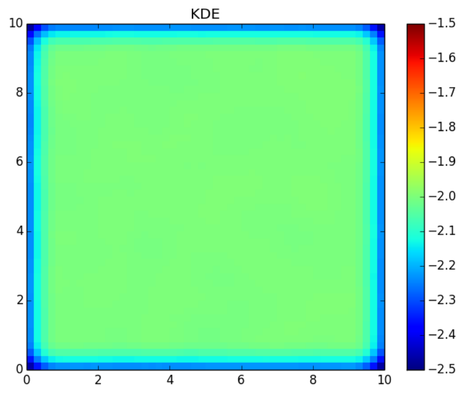
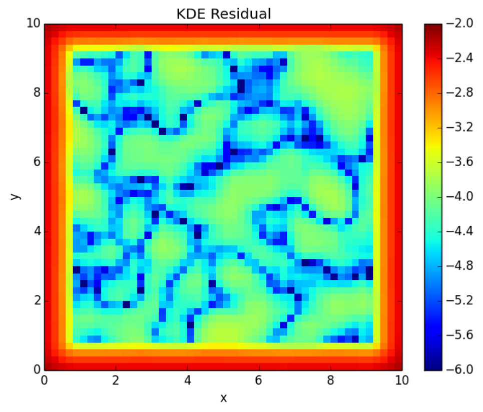


Figure 60: Residual of the scalar flux under the same conditions as Figure 59.

The residual shows that there are some cells where the tally solution approximates the analytic accurately, but there are also cells with much higher error. The behavior of the residual also indicates the tally solution to be noisy. We now use the KDE for the 2-D variation.



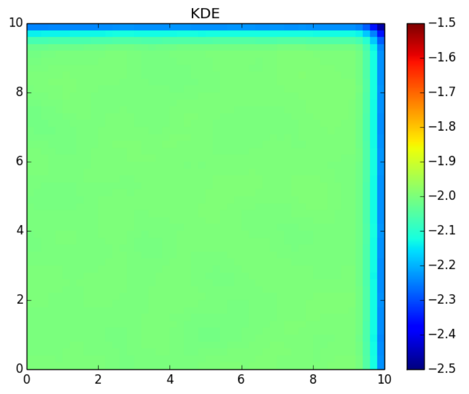
(a)



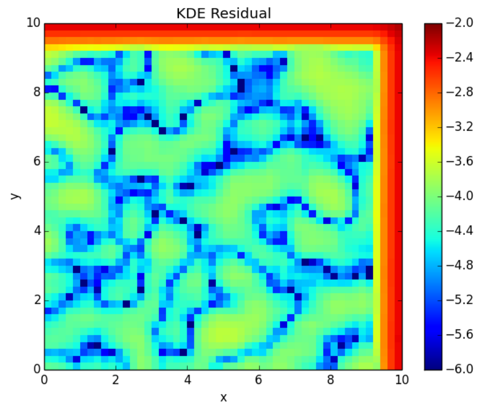
(b)

Figure 61: Kernel-density-estimated scalar flux under the same conditions as Figure 59. No boundary correction is used.

We now apply the reflective boundary correction.



(a)



(b)

Figure 62: Kernel-density-estimated scalar flux under the same conditions as Figure 59. Boundary correction is used at the origin.

With Material Feedback

Next the code is verified with material feedback where the temperature is coupled to the radiation. An infinite medium problem is used to test the code where the material is homogeneous and in thermal equilibrium with the radiation. The material and radiation temperatures should remain in thermal equilibrium throughout the entire simulation. The evolution of the material and radiation temperatures are shown in Figure 63.

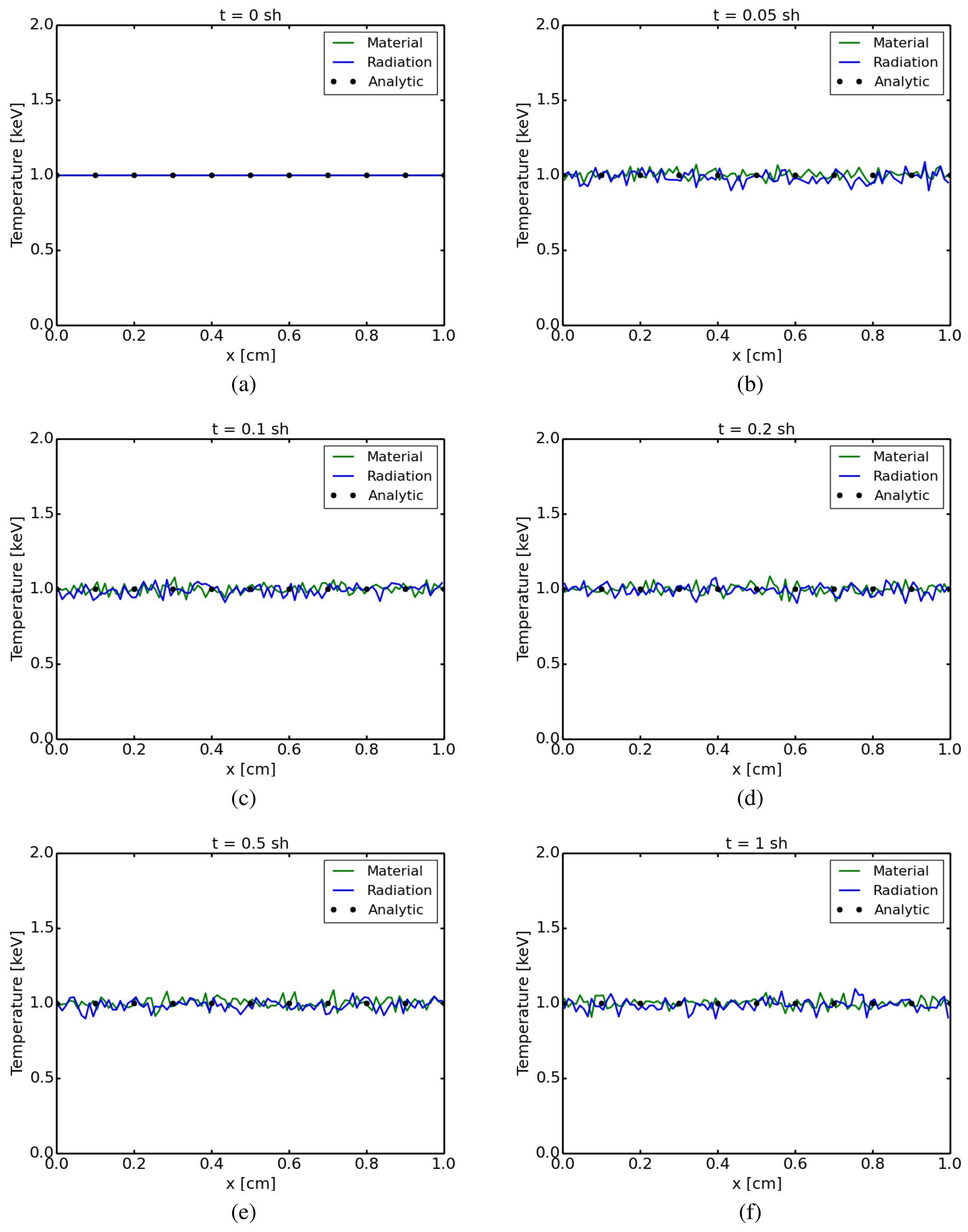


Figure 63: Material and radiation temperatures in thermal equilibrium for the infinite homogeneous medium problem. This simulation uses 100 cells, a time step of 0.01 sh, and 10^4 particles.

The temperatures of the material and radiation are noisy, but are centered around the analytic solution and remain in thermal equilibrium.

Results

We test our KDE method with an infinite medium problem and make the following assumptions:

- the material is homogeneous such that $\sigma_{\mathrm{t}}(x) = \sigma_{\mathrm{t}}$
- the material is purely absorbing: $\sigma_{\mathrm{t}} = \sigma_{\mathrm{a}}$
- the opacity has the following temperature dependence: $\sigma_{\mathrm{t}}(T) = \sigma_{\mathrm{a}}(T) = T^{-3}$
- $c_{\mathrm{v}} = 0.1$ jk/cm^{-3}
- the source is homogeneous

and apply the following initial and boundary conditions:

- $T_m^{(0)} = T_r^{(0)} = 1$ keV
- reflecting conditions on all faces

A diagram of the problem is given in Figure 64

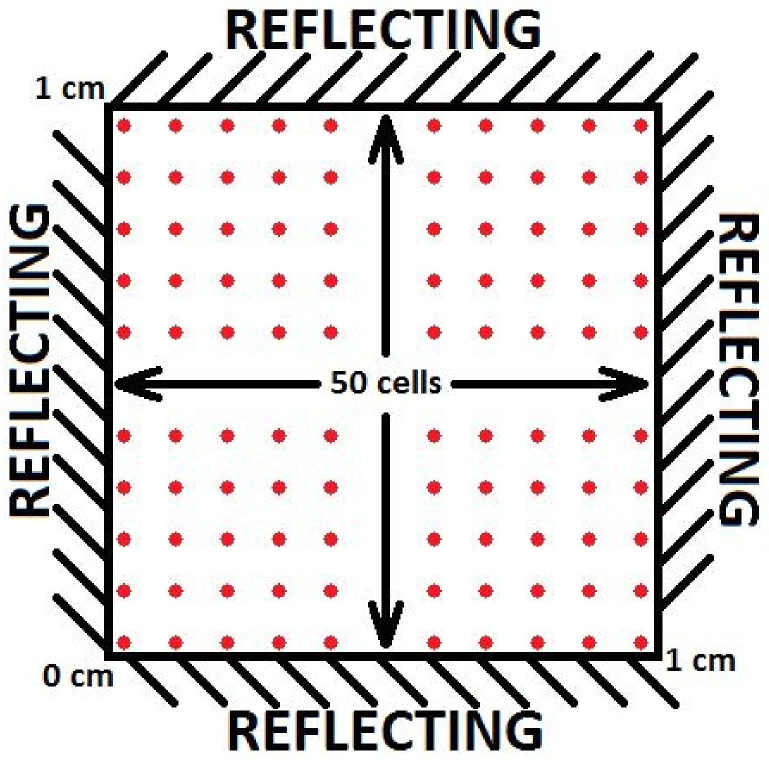


Figure 64: Diagram of the infinite homogeneous medium problem.

We first consider the 1-D case and compare the material temperatures obtained using tallies and using KDEs with reflective boundary corrections.

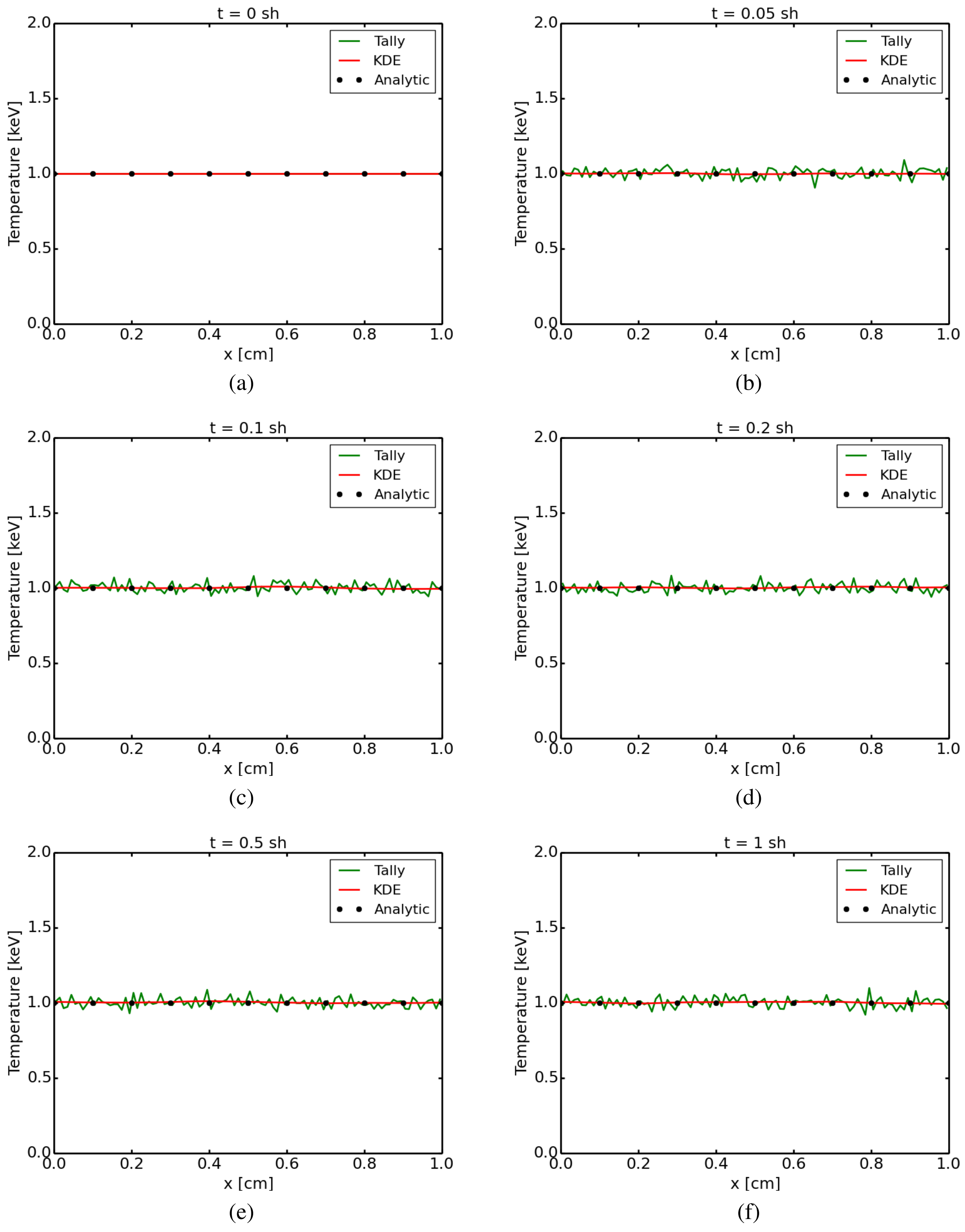


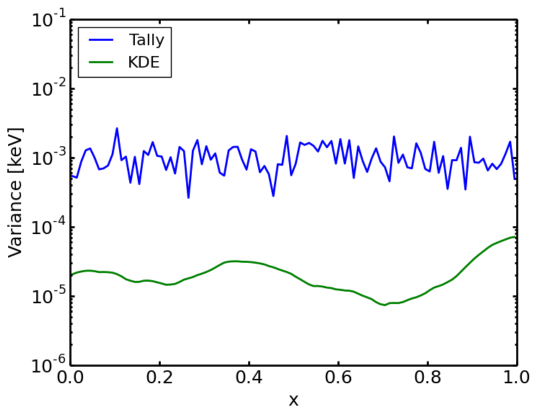
Figure 65: Material temperatures for the 1-D infinite homogeneous medium problem using 100 cells, a time step of 0.01 sh, and 10^4 particles.

The tally-estimated temperature is noisy throughout the entire simulation, while the KDE is smooth and closely approximates the analytic solution. Under these conditions, the tally solution is extremely noisy throughout the entire simulation, while the KDE solution is still smooth.

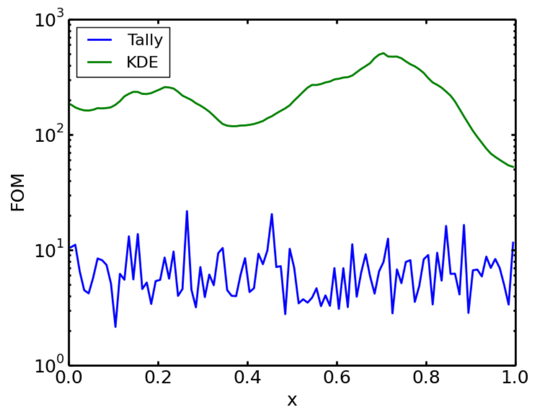
To determine the efficiency of using the KDE method, we define a figure of merit (FOM) as

$$\text{FOM} = \frac{1}{s^2 t} , \tag{59}$$

where s^2 is the variance of the solution and t is the simulation time. The variance and FOM for the infinite homogeneous medium problem with the same conditions as Figure 65 are given in Figure 66. Next we consider the 2-D case and focus on the residuals of the material temperature solutions. The residuals for the material temperature obtained using tallies and KDEs are given in Figures 67 and 68, respectively.



(a)



(b)

Figure 66: Variance and figure of merit for the infinite homogeneous problem.

The variance of the KDE is on the order of 100 times less than the variance of the tally. The cost for obtaining a reduced variance is not prohibitive, which is reflected in the FOM for using KDEs.

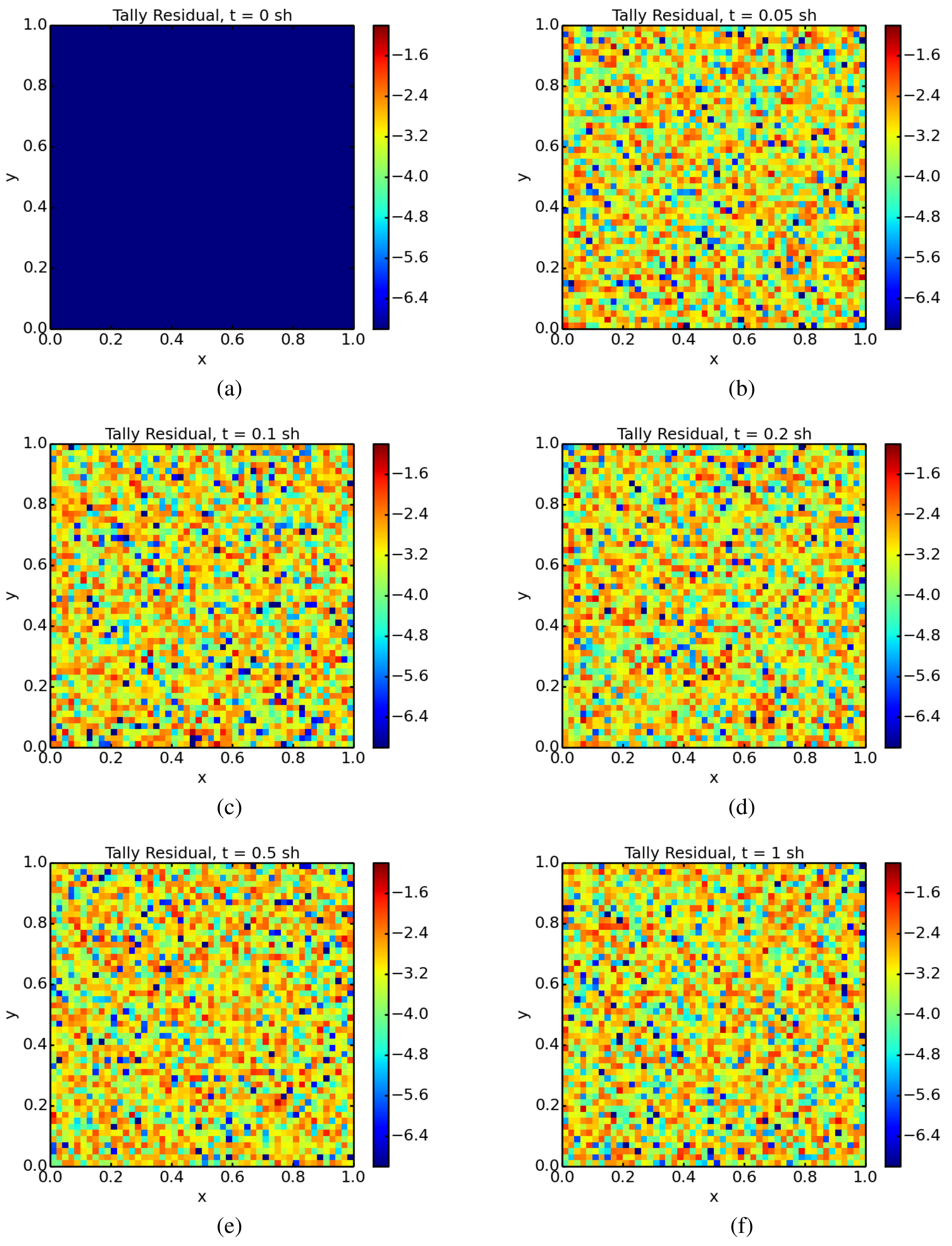


Figure 67: Tally-estimated material temperature residuals for the infinite homogeneous medium problem using 2500 cells, a time step of 0.01 sh, and 10^4 particles.

The error in the tally solution spans a large range of orders. In some cells, the approximation is accurate, while in other cells, the approximation is poor. The behavior in the residual indicates that the tally solution is noisy throughout the simulation.

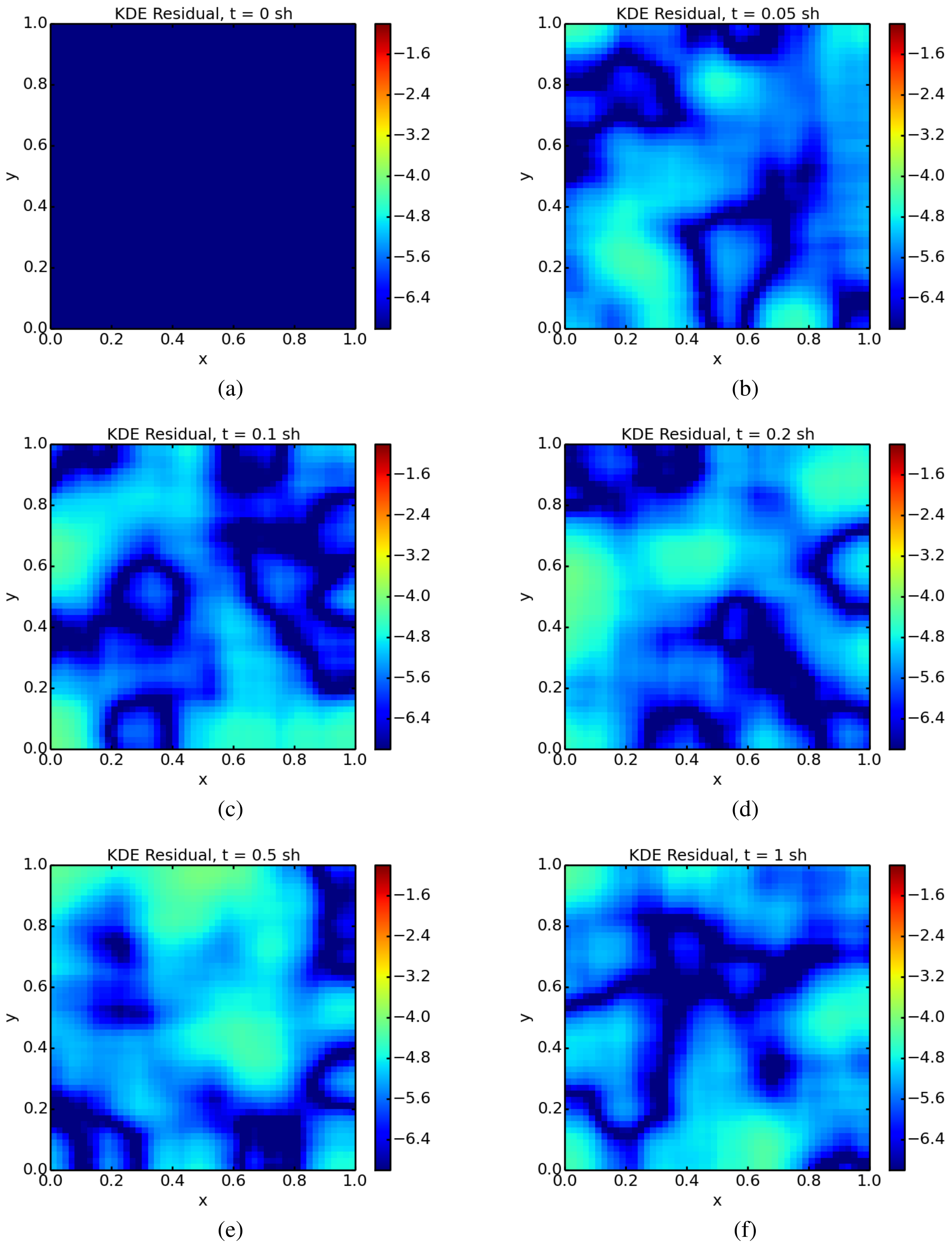
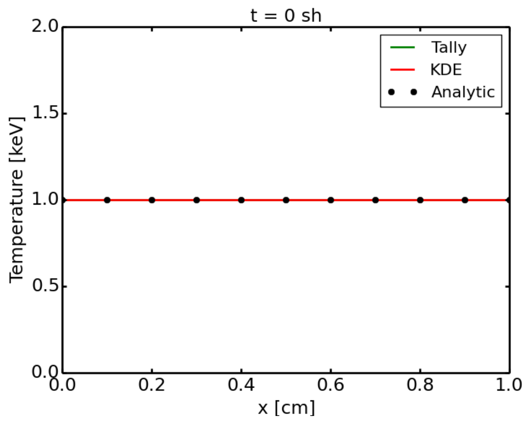
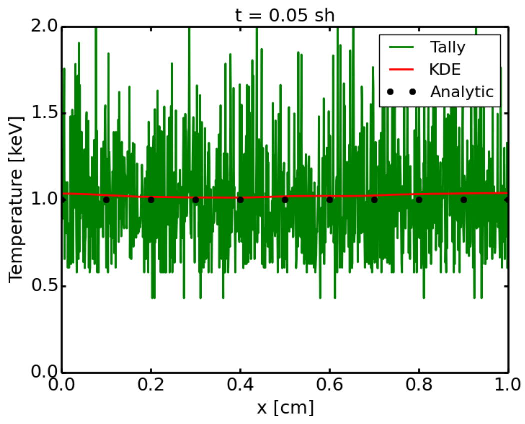


Figure 68: Kernel-density-estimated material temperatures for the infinite homogeneous medium problem using 2500 cells, a time step of 0.01 sh, and 10^4 particles.

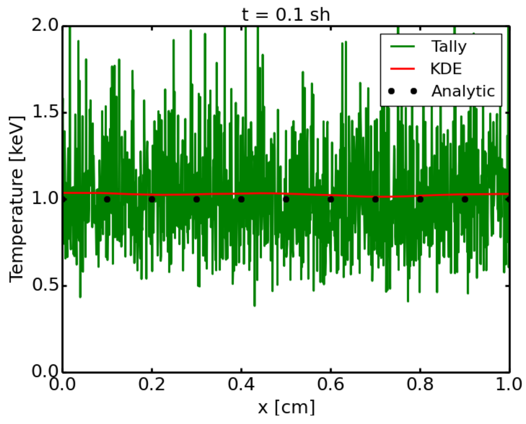
The maximum error in the KDE solution is $\sim 10^{-4}$ and has patches of accurate approximation throughout the simulation. We now consider problem parameters in which the tally-estimated temperature is extremely noisy.



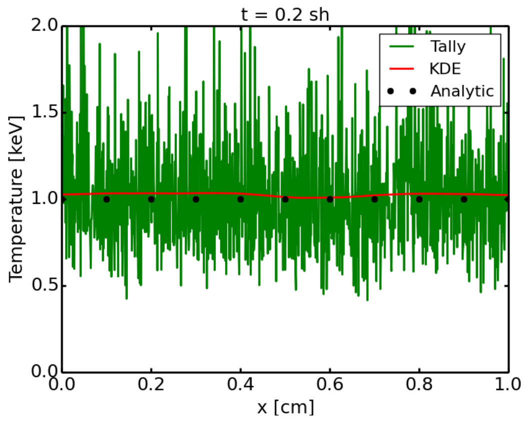
(a)



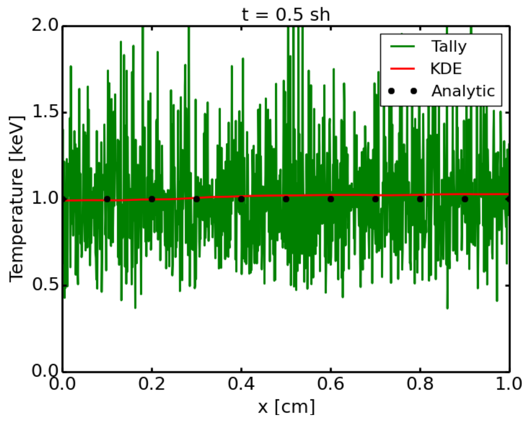
(b)



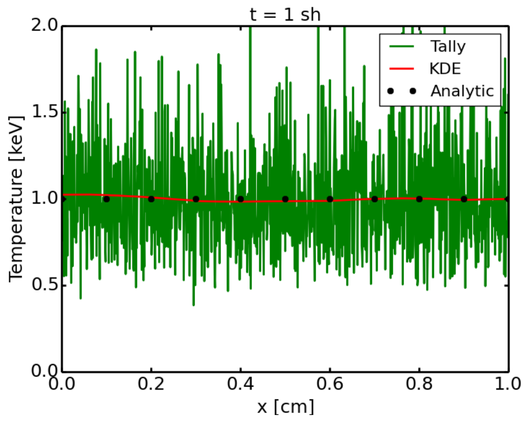
(c)



(d)



(e)



(f)

Figure 69: Material temperatures for the 1-D infinite homogeneous medium problem using 1000 cells, a time step of 0.01 sh, and 10^3 particles.

Under these conditions, the tally solution is extremely noisy throughout the entire simulation, while the KDE solution is still smooth.

Conclusion

In this work, we investigated noise reduction techniques for IMC TRT, which included source biasing and kernel density estimation. Source biasing was implemented in Milagro for the Marshak wave problem. KDEs were implemented in DESERT for the infinite homogeneous problem. Results using KDEs show that they can significantly reduce noise and that the additional computational cost is well worth the implementation. Prospects for using this method include using fewer particles in a simulation, application to meshes with adaptive refinement, and using a larger time step in problems where prohibitively small time steps are required to prevent violation of the maximum principle.

Future Work

Future work will involve coupling other noise reduction methods to KDEs as well as determining how well KDEs can resolve steep spatial and flux gradients at material interfaces. Improved boundary correction will also be investigated in order to apply KDEs to surface-source problems.

Acknowledgements

This work was performed under the auspices of Los Alamos National Laboratory under charge code JW61-0000. We would like to thank Scott Runnels for managing the Computational Physics Student Summer Workshop and all those who took part in giving the workshop lectures for their time and effort. We especially thank Mathew Cleveland, Todd Urbatsch, and Allan Wollaber for their exceptional guidance and support.

Appendix

Energy Deposition

Considering a particle traveling through a slab with length L in 1-D geometry, continuous energy deposition is used to treat the problem. As the particle travels through the slab, its energy is exponentially attenuated and the energy deposited in the material is given by

$$E_{dep} = w_0(1 - \exp(-\sigma s)) \ , \tag{60}$$

where w_0 is the particle's initial weight. We are interested in how the material's temperature is affected by the continuous energy deposition. The TRT equations in 3D geometry are given by

$$\frac{1}{c}\frac{\partial I}{\partial t} + \vec{\Omega}\cdot\nabla I + \sigma_{\mathrm{a}} I = 4\pi\sigma_{\mathrm{a}} B + Q \ , \tag{61a}$$

$$c_{\mathrm{v}}\frac{\partial T}{\partial t} = \int_0^\infty \int_{4\pi} \sigma_{\mathrm{a}}\left[I - 4\pi B\right] \, d\Omega d\nu \ . \tag{61b}$$

We assume no re-emission and no sources: $B = 0$, $Q = 0$. Integrating the LHS of (61b) and setting that equal to the energy deposition yields

$$\int_{V_{cell}} \int_{\Delta t} c_{\mathrm{v}} \frac{\partial T}{\partial t} = E_{dep} \ , \tag{62}$$

$$c_{\mathrm{v}}(T^{(1)} - T^{(0)})V_{cell} = E_{dep} \ , \tag{63}$$

$$T^{(1)} = T^{(0)} + \frac{E_{dep}}{c_{\mathrm{v}} V_{cell}} \ . \tag{64}$$

Substituting (60) into (64) yields

$$T^{(1)} = T^{(0)} + \frac{w_0(1 - \exp(-\sigma s))}{c_{\mathrm{v}} V_{cell}} \ . \tag{65}$$

Assuming a purely-absorbing slab and that the slab is discretized in the x-direction, energy deposition would decrease exponentially as the particles travel. The energy deposition, however, does not scale at the same rate as the heat capacity , $c_{\mathrm{v}} V_{\mathrm{cell}}$. As the volume of the cell decreases, the effective energy that is deposited increases dramatically, which can cause nonphysical temperature spikes. In 2-D geometry, we can express (65) as

$$T^{(1)} = T^{(0)} + \frac{w_0(1 - \exp(-\sigma s))}{c_{\mathrm{v}} \Delta x \Delta y} \ . \tag{66}$$

If the slab is discretized in the y-direction, energy deposition in each cell is equivalent, assuming a uniform radiation intensity distribution over the slab face. Refining in the y-direction would also lead to nonphysical overheating.

Plasma Transport in ICF Implosions

By Archis S. Joglekar and Mario I. Ortega, Dr. Erik L. Vold (Mentor, XCP-2)

Abstract

The effects of small-scale, atomic-level mixing and viscosity on inertial confinement plasmas currently represent challenges in ICF research. Most current ICF hydrodynamic codes ignore the effects of viscosity though research indicates viscosity and mixing have a substantial impact on neutron yields. Here we have used a one-dimensional Lagrange hydrodynamic code and implemented plasma viscosity, fuel-plastic mixing, a gray radiation diffusion approximation model, and a three temperature model for ions, electrons, and radiation to determine their impacts on neutron yield calculations. It was found that plasma viscosity has substantial impacts on ICF shock dynamics and maximum neutron production rates.

Introduction

Direct-drive inertial confinement fusion (ICF) refers to laser heating of a nearly spherical shell that contains fusion fuel, resulting in compression and fusion [64]. Perfect spherical compression has many shortcomings such as unstable to laser-plasma instabilities such as Rayleigh-Taylor and Richmeyer-Meshkov. Target non-uniformities become exaggerated as well as beam-to-beam imbalances and perfect spherical compression is compromised and degrades performance. The implosion dynamics of the capsule can be affected by the shell material mixing into the fuel.

Consequently, small scale effects on burn physics and implosions have been the focus of many studies. These include studies of shock structure [45], barodiffusion [2], flux limiters [60, 42], as well as laser absorption mechanisms [52] mostly using hydrodynamics codes. These studies present various effects that may affect implosion dynamics but none address the effect of real ion viscosity on ICF implosions. It should be noted that the replacement of Von Neumann artificial viscosity with real viscosity has been studied, [43], but our study aims to implement real viscosity in addition to artificial viscosity.

In this study, we explore the effects of viscosity on ICF implosions through the use of a one dimensional, three temperature hydrodynamics model that also includes treatment of fuel-plastic mass mixing.

Theory

Hydrodynamics

The plasma is modeled with a one-dimensional three-temperature hydrodynamics model.

$$\frac{D\rho}{Dt} = -\rho\nabla\cdot\mathbf{u} \tag{67}$$
$$\rho\frac{D\mathbf{u}}{Dt} = -\nabla p + \nabla\cdot(\eta_0\nabla\mathbf{u}) \tag{68}$$
$$\frac{n}{\gamma-1}\frac{DT}{Dt} = -p\nabla\cdot\mathbf{u} + \nabla\cdot(\kappa_e\nabla T) \tag{69}$$

Equation 67 is the conservation of density equation in the Lagrange frame while eq. 68 and eq. 69 are the conservation of momentum and energy equations, respectively. Note that the latter two equations have a diffusion term. Equation 68 has the effect of viscosity modeled through the diffusion approximation while the heat conduction in eq. 69 is performed the same way. $\eta_0 = 0.96nkT\tau$ and $\kappa_e = 3.2\frac{nkT\tau}{m_e}$ as from [29]. The plasma pressure is provided by the ideal gas approximation $p = nkT$.

The three temperature model provides for a way to more accurately describe the plasma. As such, eq. 69 is modified to include separate temperatures for the three main species in the plasma; ions, electrons, and radiation. Thermal conductivity for ions, electrons, and radiation as well as ion viscosity are strong functions of temperatures and the 3T model was implemented to determine the impact of these parameters on energy and momentum balances in the fuel and plastic zone regions.

Species Mass Mixing

In addition to the hydrodynamics, the code is also capable of tracking species mass mixing.

$$\frac{DC}{Dt} = -\nabla\cdot\left[\frac{p_{ia}\rho_j}{\nu_{ij}\rho}\left(\nabla\chi_i - p_{ia}^{-1}\left[(\chi_i - C_i)\nabla p_{ia} - (Z_i - C_i)\nabla p_e\right]\right)\right] \tag{70}$$
$$-\nabla\cdot\left[\frac{p_{ia}\rho_j}{\nu_{ij}\rho}\left(-p_{ia}^{-1}\left[(Z_i - C_i)F_{T_e} + (F_{T_i} - Y_iF_{T_{ia}})\right]\right)\right] \tag{71}$$

where, C is the mass fraction of the light species, p_{ia}, is the total ion pressure, ρ_j is the mass density of the heavier species. ν_{ij} is the collisional frequency between the two species. χ_i is the number fraction of the lighter species, Z_i is the charge fraction. The first three terms, the molar gradient, and the ion barodiffusion and electron barodiffusion terms are well known and defined but the terms for the thermal mixing forces have not been well defined in literature.

$$F_T = -\frac{3}{2}\left(\frac{\nu_{ij}}{\nu_{ii}+\nu_{ij}}n_i\nabla T_i + \frac{\nu_{ei}(1-Z)+Z\nu_{ej}}{\nu_{ee}+\nu_{ei}+\nu_{ej}}n_e\nabla T_e\right) \tag{72}$$

So for those terms not well defined in eq. 71, the code uses the formulation given in eq. 72. The collision frequencies are given by the relations in [49].

Three Temperature Model for Ions, Electrons, and Radiation

The deuterium-deteurium nuclear fusion reactions and the temperatures of ion, electrons, and radiation were modeled by four coupled ordinary differential equations:

$$\frac{dN_D}{dt} = -\frac{1}{4}N_D^2 < \sigma v > \tag{73}$$

$$\frac{n_i}{\gamma - 1}k\frac{dT_i}{dt} = -(\Pi + q + \eta\nabla\vec{u})\nabla\cdot\vec{u} + \nabla\cdot(\kappa_i\nabla T_i) + E_pR_{DD}f_{pi} + \omega_{ie}(T_e - T_i) \tag{74}$$

$$\frac{n_e}{\gamma - 1}k\frac{dT_e}{dt} = -p_e\nabla\cdot\vec{u} + \nabla\cdot(\kappa_e\nabla T_e) + E_pR_{DD}f_{pe} + \omega_{ie}(T_i - T_e) + (\omega_p + \omega_c)(T_r - T_e) + S_{laser} \tag{75}$$

$$\frac{16\sigma T_r^3}{c}\frac{dT_r}{dt} = -p_r\nabla\cdot\vec{u} + \nabla\cdot(\kappa_r\nabla T_r) + (\omega_p + \omega c)(T_e - T_r) \tag{76}$$

where N_D is the deuterium number density, T_i, T_e, and T_r are the ion, electron, and radiation temperatures respectively. Heat sources in each equation consist of energy deposition due to the laser S_{laser}, nuclear fusion particle energy deposition E_p, heat conduction $k\nabla T$, non-adiabatic compression heating from the ion pressure Π, artificial viscosity q, and viscous heating $\eta\nabla\vec{u}$.

Ion-Electron and Radiation Coupling

Ion-electron and electron-radiation coupling terms represent energy exchange between the two different species and radiation [18]. Energy is exchanged through ion-electron collisions and electron-photon Compton and Planck scattering.

Ion-Electron Coupling

The ion-electron coupling term characterizes Coulomb collisions during the deuterium-deuterium burn. The coupling coefficient assumes ion and electron distributions are near Maxwellian [70]. Free electrons in plasma are deflected by ions (deuterium, helium-3, or protons). The energy exchange can be characterized by an energy equilibration time:

$$\tau_{ie} = \frac{3(2\pi^3)^{1/2}\varepsilon_0^2 m_i m_e}{n_e q_i^2 q_e^2 L}\left(\frac{kT_e}{m_e} + \frac{kT_i}{m_i}\right)^{3/2} \tag{77}$$

where ε_0 is the permittivity of free space, m_i and m_e are the masses of the ions and electrons respectively, q_i and q_e are the charges of the ions and electrons, and L is the Coulomb logarithm.

The ion-electron coupling coefficient ω_{ie} is then given by

$$\omega_{ie} = \frac{n_i}{\tau_{ie}} \tag{78}$$

where n_i is the ion number density.

Radiation Coupling-Compton Scattering

The Compton scattering coupling term ω_c represents the transfer of energy from radiation to electrons through photon-free electron collisions. The coupling term is given by:

$$\omega_c = \frac{16\sigma_T n_e \sigma}{m_e c^2} T_r^4 \tag{79}$$

where σ_T is the Thomson cross-section, σ is the Stefan-Boltzann constant, m_e is the mass of the electron, n_e is the atomic density of electrons, and c is the speed of light.

Radiation Coupling-Planck Opacity

The Planck opacity coupling coefficient ω_p determines the rate at which radiation couples to the ions and electrons in the system. A power fit function determined using the Los Alamos National Laboratory TOPS tool was used to determine Planck opacity values K_p for the deuterium-tritium fusion reaction and assumed valid for the deuterium-deuterium fusion reaction simulation [70].

The Planck coupling coefficient is given calculated as:

$$\omega_p = 4\rho K_p \sigma (T_e^2 + T_r^2)(T_e + T_r). \tag{80}$$

The Planck opacity coupling coefficient equation is the linearized coupling between black-body energy and radiation energy.

Numerical Methodology

The numerical scheme is as follows. The mass and momentum equations are evaluated to calculate the new density and position of each Lagrangian co-ordinate.

$$q_i = \begin{cases} \rho_i |\Delta u_i| (c_Q |\Delta u_i| + c_L c_{s,i}), & \text{if } \Delta u_i < 0 \\ 0, & \text{otherwise,} \end{cases} \tag{81}$$

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{\rho \Delta r}((P_{\text{all}} + q)_i^n - (P_{\text{all}} + q)_{i-1}^n) \tag{82}$$

$$u_{\text{all}}^{n+1} = \nabla \cdot \eta_0 \frac{\Delta u_{\text{all}}^n}{\Delta r} \tag{83}$$

$$r_i^{n+1} = r_i^n - \frac{1}{2}\Delta t (u_i^{n+1} + u_i^n) \tag{84}$$

$$\rho_i^{n+1} = \rho_i^n \frac{V_i^n}{V_i^{n+1}} \tag{85}$$

$$\tag{86}$$

The divergence operator is differenced spherically as given in eq. 87. In case of a quantity that is not located on the interfaces, and rather located on the zones, eq. 88 is used to calculate the quantity at the interface.

$$\nabla \cdot X_i = \frac{1}{dr_i} \frac{1}{[0.5(r_i + r_{i+1})]^2} \left(r_{i+1}^2 X_+ - r_i^2 X_- \right) \tag{87}$$

$$X_\pm = 2.0 \left(X_{i\pm1}^{-1} + X_i^{-1} \right)^{-1} \tag{88}$$

Then, the three temperature equations are solved, that correspond to eq. 76, for the work performed by pressure of each species, and the diffusion approximation for the heat conduction of each species.

$$Ti_i^{n+1} = Ti_i^n - \frac{dt}{dr_i} (\gamma - 1) \frac{\mathrm{Vol}_i^{n+1}}{N_{i \text{ ions}}} \frac{0.5Pi_i + q_i + \eta_0 (u_i^{n+1} - u_{i-1}^{n+1})/dr_i^{n+1}}{r_{i+1}^2 u_{i+1}^{n+1} - r_i^2 u_i^{n+1}} \tag{89}$$

$$Te_i^{n+1} = Te_i^n - \frac{dt}{dr_i} (\gamma - 1) \frac{\mathrm{Vol}_i^{n+1}}{N_{i \text{ electrons}}} \frac{0.5Pe_i}{r_{i+1}^2 u_{i+1}^{n+1} - r_i^2 u_i^{n+1}} \tag{90}$$

$$Tr_i^{n+1} = Tr_i^n - \frac{dt}{dr_i} \frac{c}{16\sigma_{SB} Tr_i^3} \frac{0.5Pr_i}{r_{i+1}^2 u_{i+1}^{n+1} - r_i^2 u_i^{n+1}} \tag{91}$$

$$Ti_{\text{all}}^{n+1} = \nabla \cdot \kappa_i \frac{\Delta Ti_{\text{all}}^n}{\Delta r} \tag{92}$$

$$Te_{\text{all}}^{n+1} = \nabla \cdot \kappa_e \frac{\Delta Te_{\text{all}}^n}{\Delta r} \tag{93}$$

$$Tr_{\text{all}}^{n+1} = \nabla \cdot \kappa_r \frac{\Delta Tr_{\text{all}}^n}{\Delta r} \tag{94}$$

The implicit computations, eq. 83, 92, 93, 94 are performed using the Thomas algorithm for solving a tridiagonal matrix. The tridiagonal is formed from the implicit differencing of the diffusion equation.

Following eq. 94, the three temperature coupling is performed through an iterative procedure that converges to a relative error of an eV i.e. $10^{-3} \geq \max(1 - Ti_{n-1}/Ti_n, 1 - Te_{n-1}/Te_n, 1 - Tr_{n-1}/Tr_n)$ with the coupling equations given in eq. 76 in forward time differenced form.

After the three temperature equations are evaluated, the species mass mixing algorithm is executed. The code has the option of performing mass species mixing implicitly by only using the molar gradient term in eq. 71 or explicitly if any other terms in eq. 71 are desired. The diffusion coefficient, $\frac{p_{ia}\rho_j}{\nu_{ij}\rho}$ is evaluated by computing the harmonic mean for the value at the interface between the two zones of interest. The dr factor due to the gradients in each term is also included in the harmonic mean.

$$\left(\frac{p_{ia}\rho_j}{\nu_{ij}\rho} \frac{1}{dr} \right)_\pm = 2.0 \left(\left(\frac{p_{ia}\rho_j}{\nu_{ij}\rho} \frac{1}{dr} \right)_{i\pm1}^{-1} + \left(\frac{p_{ia}\rho_j}{\nu_{ij}\rho} \frac{1}{dr} \right)_i^{-1} \right)^{-1} \tag{95}$$

It is important to note that ρ_j/ν_{ij} results in the cancellation of the density of the heavier species, n_j, and thus avoiding dividing by 0. The gradient terms are calculated in the following fashion,

$$\nabla\chi = \chi_i - \chi_{i-1} \tag{96}$$

The final part of the hydrodynamics, and a single time step, requires the computation of the different pressures for the ions, electrons, and radiation, P_i, P_e, P_r. These are given as function of the mix and temperature.

$$Pi_i^{n+1} = (nDD_i^{n+1} + nCH_i^{n+1})k_B Ti_i^{n+1} \tag{97}$$
$$Pe_i^{n+1} = ne_i^{n+1} k_B Te_i^{n+1} \tag{98}$$
$$Pr_i^{n+1} = 4.0/3.0(\sigma_{SB}/c)(Tr_i^{n+1})^4 \tag{99}$$

Note that the temperature in the code is in units of Joules while the above relation requires temperature units. Unit conversions are very pervasive. nDD_i and nCH_i refer to the ion number densities of DD and CH respectively. The electron density is a function of the ionization fraction and ion density at the specified grid point. $n_e = (Z_{DD}nDD_i) + (Z_{CH}nCH_i)$ where Z is the ionization fraction that is set to 1 when the electron temperature reaches the first ionization potential for Hydrogen.

Qualitative Validation of Mass Mixing Algorithm

The mass species mixing algorithm used [45] as a guideline. The goal was to be able to qualitatively replicate the results in this paper as part of the mass mixing abilities of this code.

Namely, [45] shows that the heavier material has a sharper density gradient while the lighter material diffuses in an approximately gaussian shape into the heavier material. Figure 70 shows the results from their calculation for a system in pressure equilbrium that contains Deuterium on one side and Gold on the other.

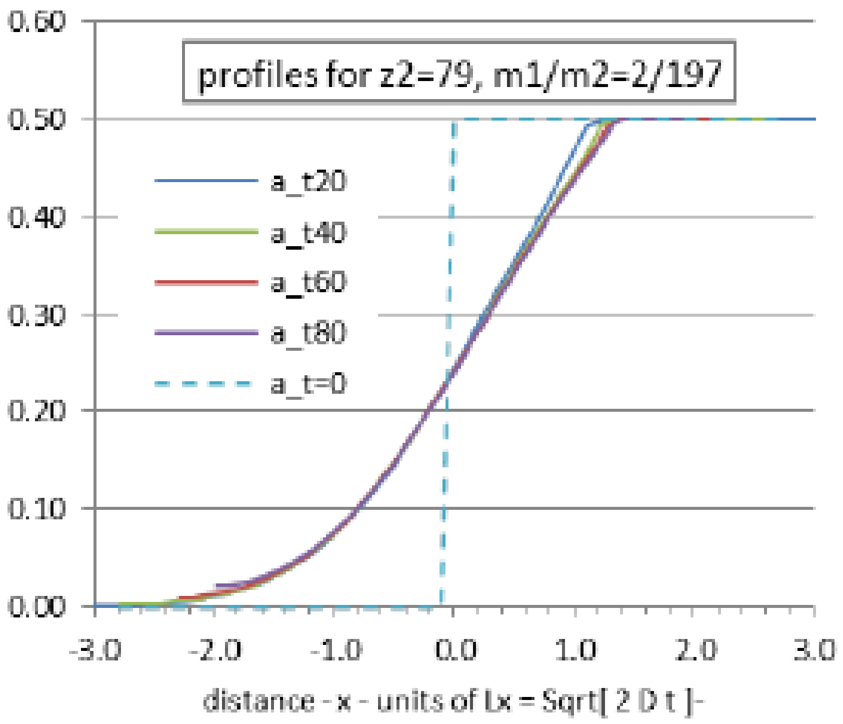
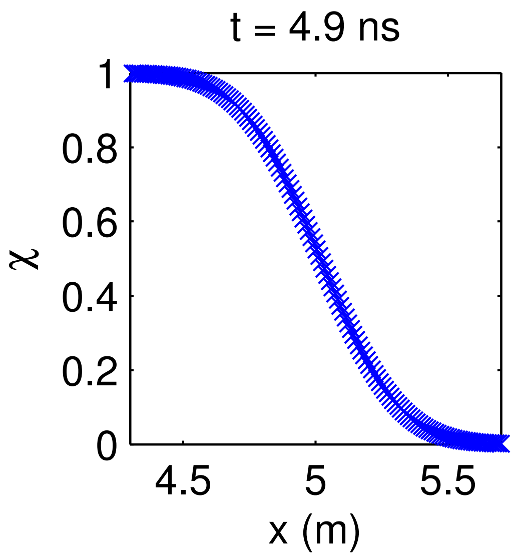
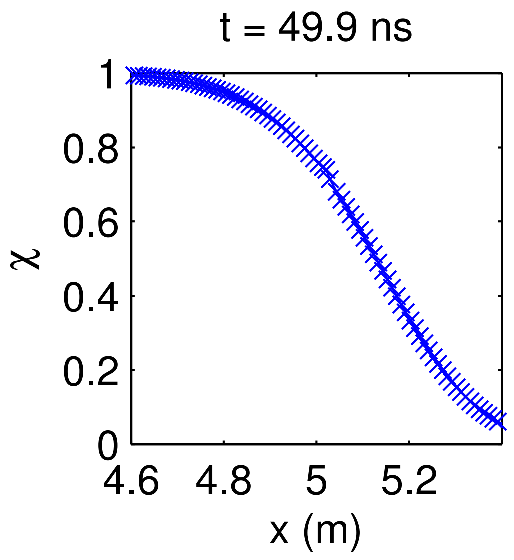


Figure 70: Time dependent numerical solutions for mixing problem [45]. The metal is on the right side ($x > 0$) while the gasis on the other ($x < 0$).

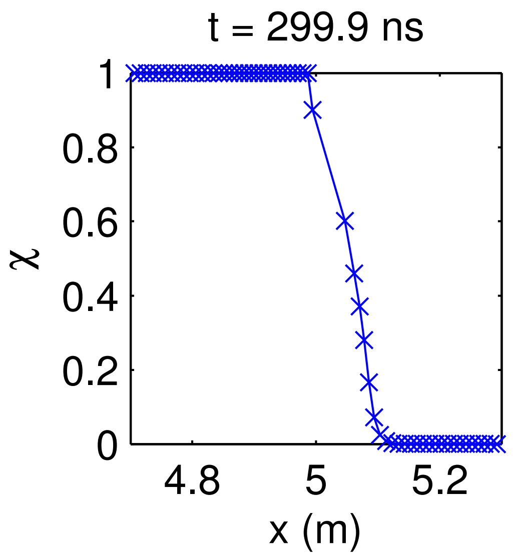
In order to compare, simulations initialized with a pressure equilibrium were performed for three separate material combinations. Deuterium-Deuterium, Deuterium+Deuterium-CH, Deuterium-Gold. Results from these are shown in fig. 71



(a) DD-DD mass mixing test



(b) DD-CH mass mixing test



(c) D-Au mass mixing test

Figure 71: Mass mixing tests. In particular (c) is to be compared to fig. 70

Figure 71 qualitatively indicates that the mass mixing is in accordance with [45]. Figure 71(a) shows symmetry as well as significant mixing at 4.9 ns. Since Deuterium has low atomic mass it has a large diffusion coefficient. Therefore, mass mixing does not require a long time, in contrast to figs. 71(b,c).

Figure 71(a) is a similar plot but for a DD-CH interface. It exhibits asymmetry at the mixing layer where the inflection point is moved further up in χ. This suggests that the fuel, or gas, moves freely into the plastic. Note that this effect takes 49.9 ns to manifest, and is still not very clear. Figure 71(c) exhibits this property clearly, where the Deuterium enters the Gold along a smooth gaussian like shape but not much Gold enters the Deuterium region. This period of evolution is much larger, 299.9 ns.

Results

The addition of plasma physics models necessitated extensive testing. A base case, described in the next subsection, was selected and used to validate the code models and algorithms. In addition to the base case, various cases were run using different physics models. Results for these runs using the 3T model, mass species mixing, viscous and inviscid situations, and flux limiter models are described in the following sections.

Three Temperature Model Results for DD-CH implosion

Table 3: ICF Base Case Parameters

Hydrodynamics Model	1-D Staggered Grid Lagrangian
Viscosity Effects	Momentum diffusion and viscous heating of ions
Species Mixing	Molar, electron, and ion pressure and temperature gradients
Temperature Model	3T Model for Ions, Electrons, and Radiation
Transport Coefficients	Naval Research Laboratory Formulary
Radiation Model	Gray radiation diffusion approximation
Laser Heating	Uniform deposition across a specified width

A base case was selected to compare with later runs. The different parameters were varied in order to gauge the impact of physics such as viscosity and mass mixing. Table shows the base case parameters selected for this project.

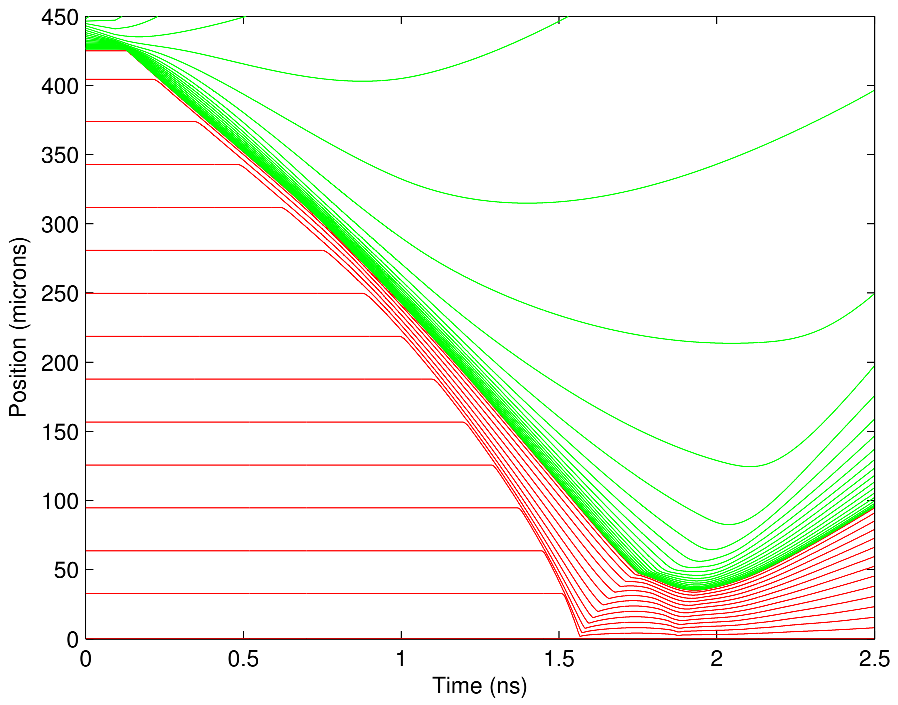


Figure 72: Radius-Time Plot for ICF Base Case

A radius-time plot of the Lagrangian fuel and plastic zones for the base case is shown in fig. 72. Each line represents a Lagrangian coordinate, a fixed point along the fuel capsule. The movement of these lines shows the movement and compression of the fuel capsules. Laser heating of the plastic zones can be seen at the beginning of the simulation and the compression of the fuel continues until it is approximately 35 microns wide. The shock arrives at approximately 1.56 ns and the return shock arrives at approximately 1.85 ns.

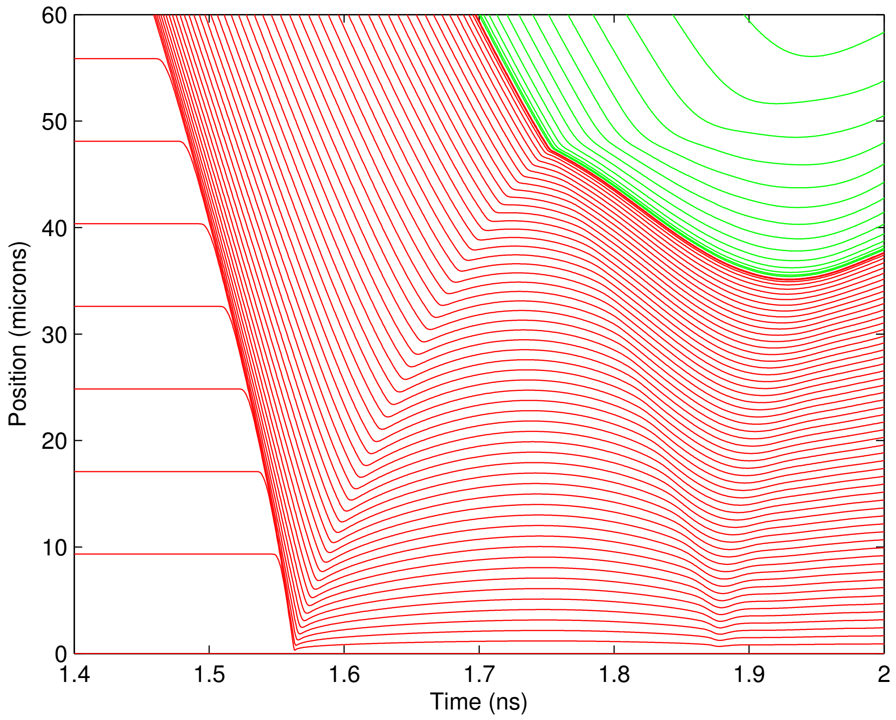
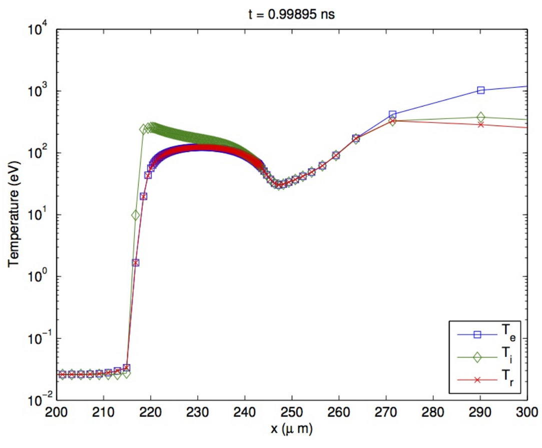
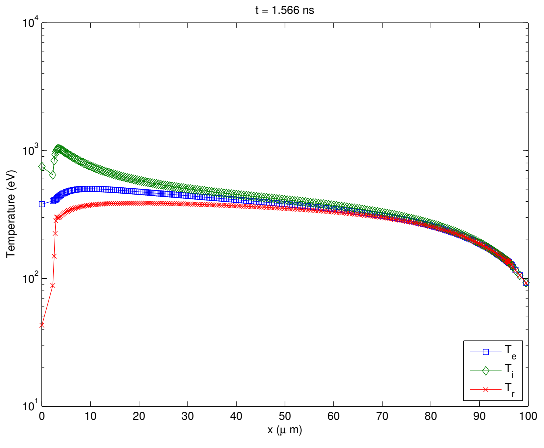


Figure 73: Magnified Radius-Time Plot for ICF Base Case

Figure 73 shows a zoomed in view of the radius-time plot, where the shock dynamics in the fuel during compression are illustrated.



(a) (a)



(b) (b)

Figure 74: Ion, Electron, and Radiation Temperatures (a) Prior to Laser Shutoff, (b) at time of incidence of first shock.

Inclusion of the 3T model for ion, electron, and radiation temperatures allowed for shock ion heating, electron laser heating, and other physical phenomena to be observed in the 1D code. Laser energy is deposited in electrons and this can be seen in fig. 74 where the electrons are the hottest near the end of the fuel capsule at high radius. Also evident is the heating of ions due to the shock front and the closely coupled nature of the electron and radiation temperatures. Electron thermal conductivity diffuses the shock front which is very evident for the ions. Since the thermal conductivity for the ions is smaller by a factor of $\sqrt{m_e/m_i}$, the ion shock front lacks the diffusive characteristic. Note that due to the strong electron heat conduction, the temperature a few microns in front of the shock front is also slightly elevated.

Note the separation in temperatures at the radial center resulting from the uncoupling. This could be due to shock incidence where the strong ion pressure front causes heating of the ions at the center, while the radiation temperature is the smallest due to the relative lack of radiation pressure.

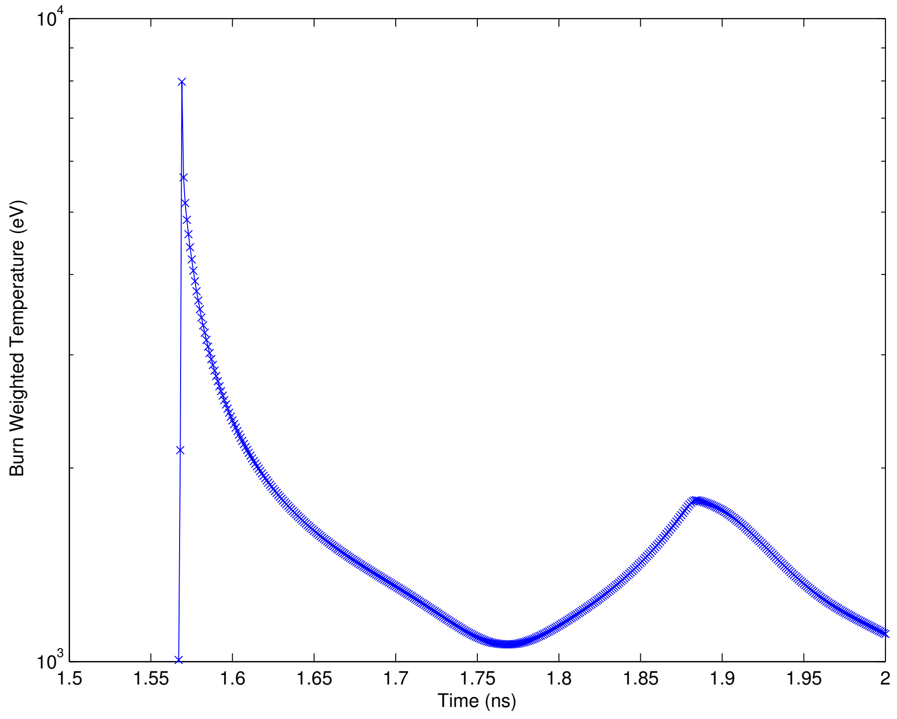


Figure 75: Burn Weighted Ion Temperature

The calculation of burn weighted ion temperature, illustrated in fig. 75 was used as a diagnostic to compare the code to experimental and other numerical results. The burn weighted ion temperature in the base case was found to be approximately 7 keV, which compares well to experimental results and other 1D ICF codes [25, 17].

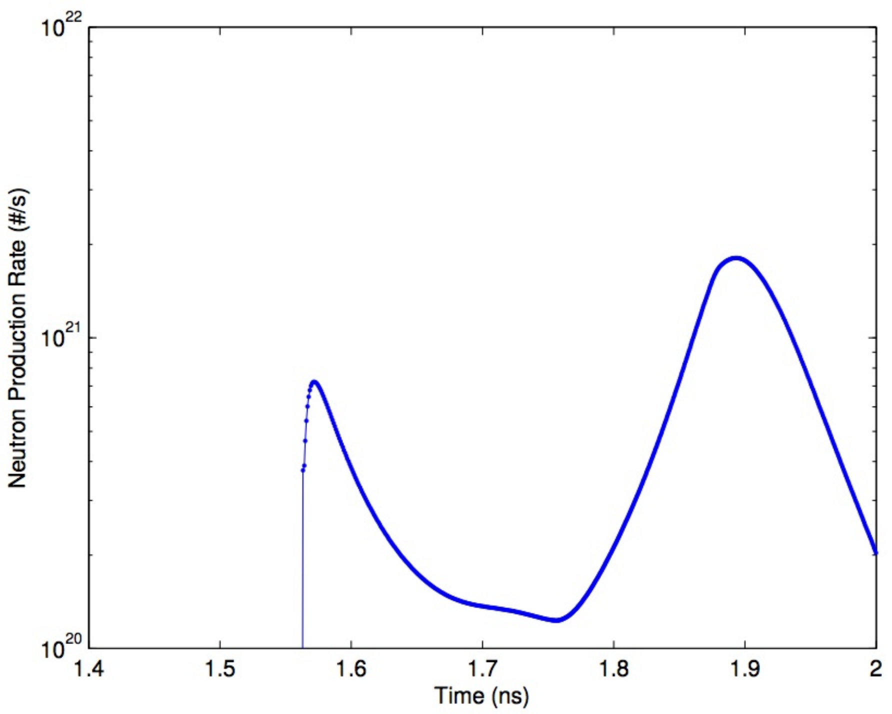


Figure 76: Neutron Production Rate

The neutron production rate over time, shown in fig. 76, were also calculated for comparison with experiment and simulation. For the base case, neutron production rates exhibit two local maximum values, corresponding to the times of shock heating and compression burn. Maximum neutron production rates were found to be approximately at 1.86 ns, which agree well with [25, 17].

Results of Comparison between Inviscid and Viscous Plasma Transport

The radius-time plot of the inviscid case overlaid onto fig. 72 the radius-time plot from the viscous case illustrates the difference in timings between the two shocks.

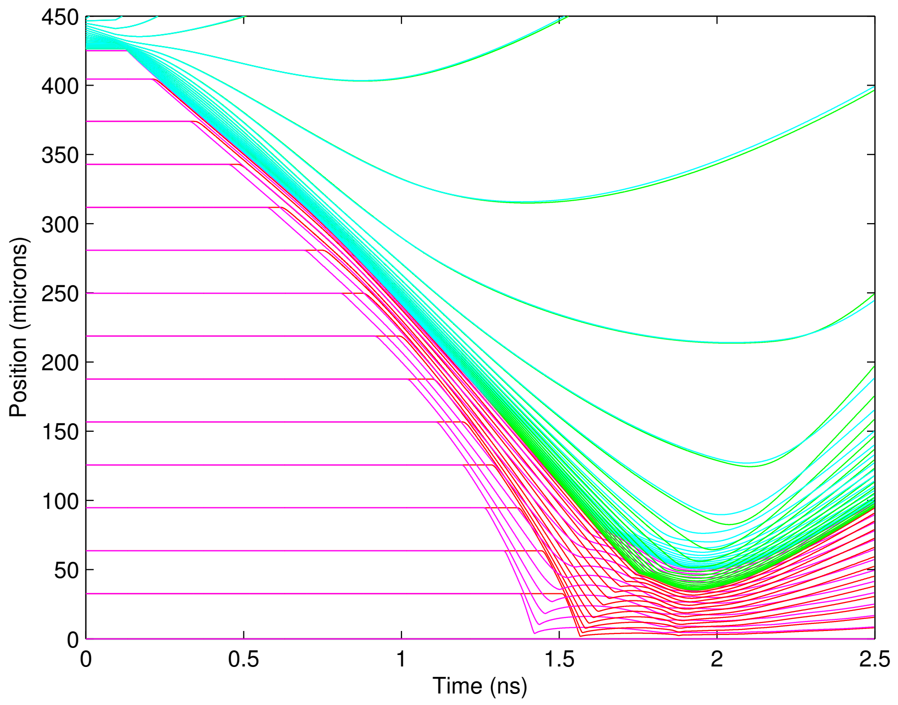


Figure 77: Radius-Time plot of the inviscid and viscous case. The Pink and Cyan colors refer to Fuel and Plastic, respectively, for the inviscid case and Red and Green refer to the same for the viscous case.

Figure 77 suggests that viscous effects do not affect the plastic. The Lagrangian coordinates show strong agreement by overlapping in the green and cyan case. This is because the deposited laser energy quickly moves into the fuel and the plastic does not retain a high temperature for very long. Therefore, the strong temperature dependence of the ion viscosity is not accentuated in the plastic.

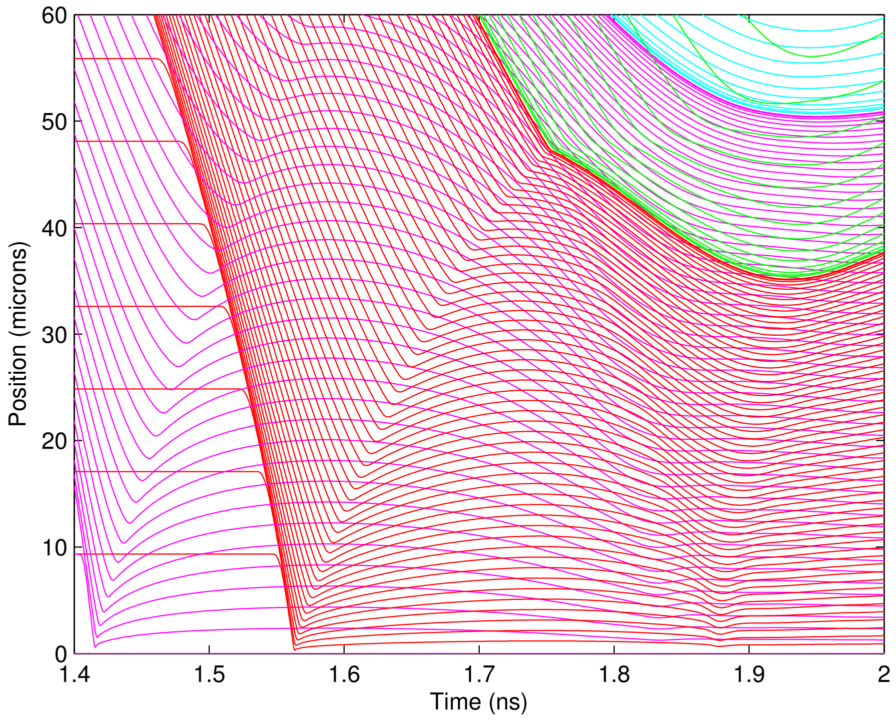


Figure 78: Radius-Time plot of the inviscid and viscous case. The Pink and Cyan colors refer to Fuel and Plastic, respectively, for the inviscid case and Red and Green refer to the same for the viscous case.

Meanwhile, the fuel behaves very differently for both cases. The timing of the first shock is visibly different. A better illustration of the compression dynamics during this time are more evident in fig. 78. The difference in the timing of incidence of the first shock is approximately 0.2 ns. As mentioned previously, the viscous shock arrives at 1.56 ns and returns at 1.86 ns. Meanwhile, the inviscid shock arrives at 1.41 ns and returns 0.45 ns later, a difference of a factor of 150%. Aside from the shock timings, fig. 78 shows that the fuel reaches a minimum radius of approximately 40 μm for the viscous case in comparison to an $r_{\text{min}} = 60\ \mu$m in the inviscid case. The distance between the lines representing Lagrangian coordinates is larger for the inviscid case suggesting that the compression of each zone is smaller in magnitude. This corresponds to a smaller maximum density for the inviscid simulation. Figure 78 also shows that the second shock comes in at a larger velocity in the inviscid case, as suggested by the slope of the lines representing the returning shock.

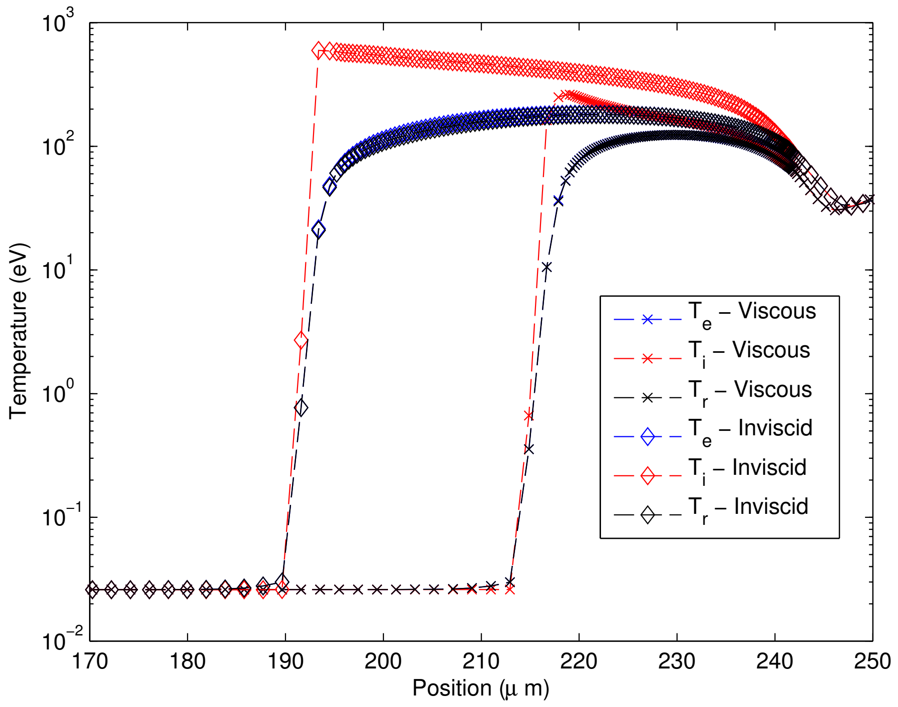


Figure 79: Electron, Ion, and Radiation Temperature for Inviscid and Viscous cases at $t = 0.99$ ns.

A snapshot of the three temperature profiles at 0.99 ns, are shown in fig. 79. The shock front for the viscous case is a full 20 μm behind that from the inviscid case. This is a result of the shocked region being larger by that distance, as the rear end of the shocked region is near the same radial position in both cases. Additionally, the temperatures for all three species is much higher in the inviscid case due to the lack of viscous dissipation. This also shows in the form of a much steeper ion shock front as well as a clear exponential decay. The snapshot from the viscous simulation shows a slight bump to the right of the shock front as well as a much smoother front. This slight bump is in direct contrast with the clear exponential profile of the inviscid shock. The bump is caused because the viscous case allows for compression and heating due to viscous effects at the location of the shock front. The shock front provides for a good candidate for this effect due to the high temperature resulting in a large viscosity and the velocity gradient that is indicative of the shock.

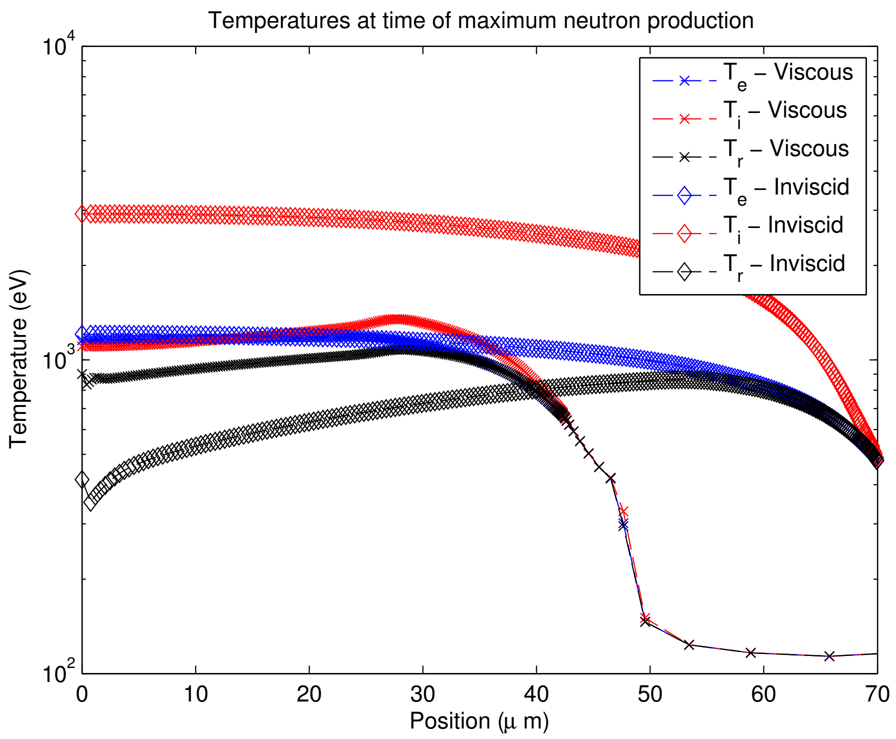


Figure 80: Electron, Ion, and Radiation Temperature for Inviscid and Viscous cases at time of maximum compression and neutron production.

Figure 80 is a similar plot but at the time of maximum neutron production. This occurs at different physical times for both cases due to the differences in timing caused by viscous effects. Regardless, this figure shows that at maximum compression, the shocked region is approximately 20 μm larger in the inviscid case, as also inferred from fig. 78 and fig. 79. This suggests that the maximum compression, r_{min}, is a function of the length of the shocked region.

It can also be seen that the maximum ion temperature is larger by a factor of 3 in the inviscid case at this time due to the lack of viscous momentum dissipation. The electron temperatures are within 10 eV of each other in the two cases. This could be because the viscous effects are not included for the electron population, due to the mass dependence in the viscous transport coefficient and consequently, the energetics are very similar. The radiation temperatures are different by a factor of 2, the inviscid case being close to 500 eV while that from the viscous simulation shows approximately 1 keV.

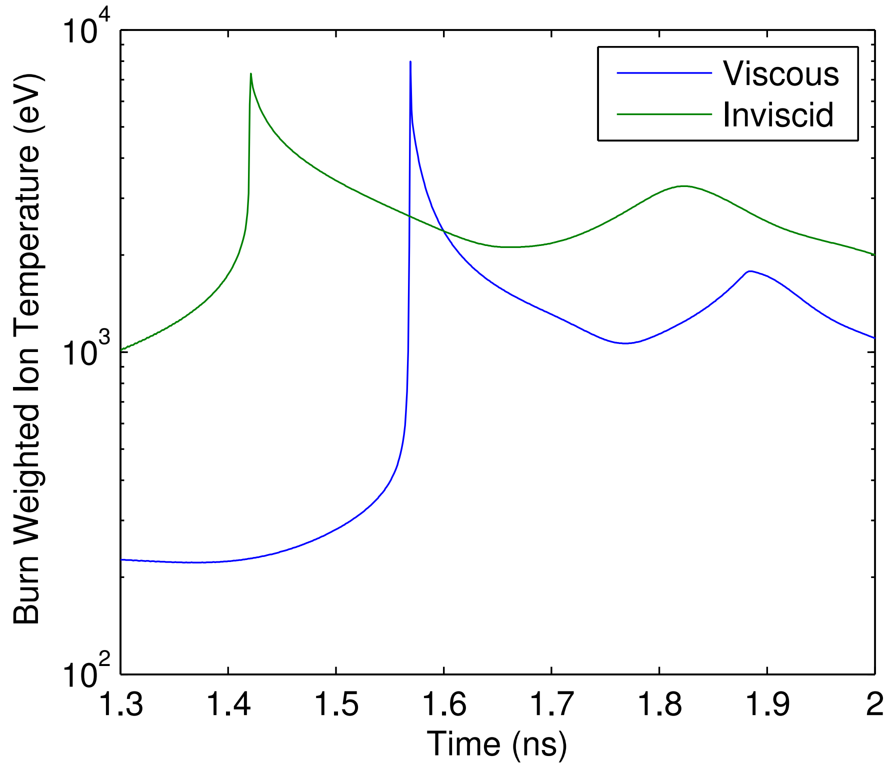


Figure 81: Burn weighted ion temperature over time for both cases. The temperature is consistently higher in the inviscid case, as expected.

Figure 81 shows that both simulations match previous calculations and measurements of the maximum burn weighted ion temperature at 7 keV. The difference in the timing of the arrival of the first shock is also evident. The viscous case undergoes a larger drop in temperature as indicated by the steep gradient after the arrival of the first shock. The second peak in the burn weighted ion temperature is smaller by a factor of 3. This figure also shows the discrepancy in the time of compression burn for both simulations. Since the compression burn temperatures are so different, one can expect a discrepancy in the neutron production rate as well.

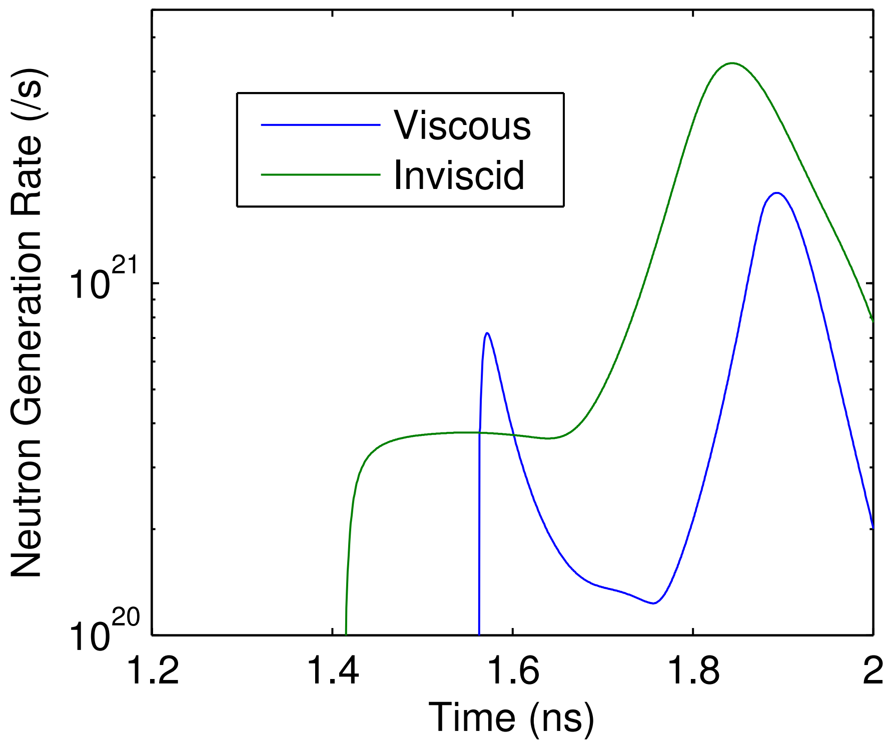
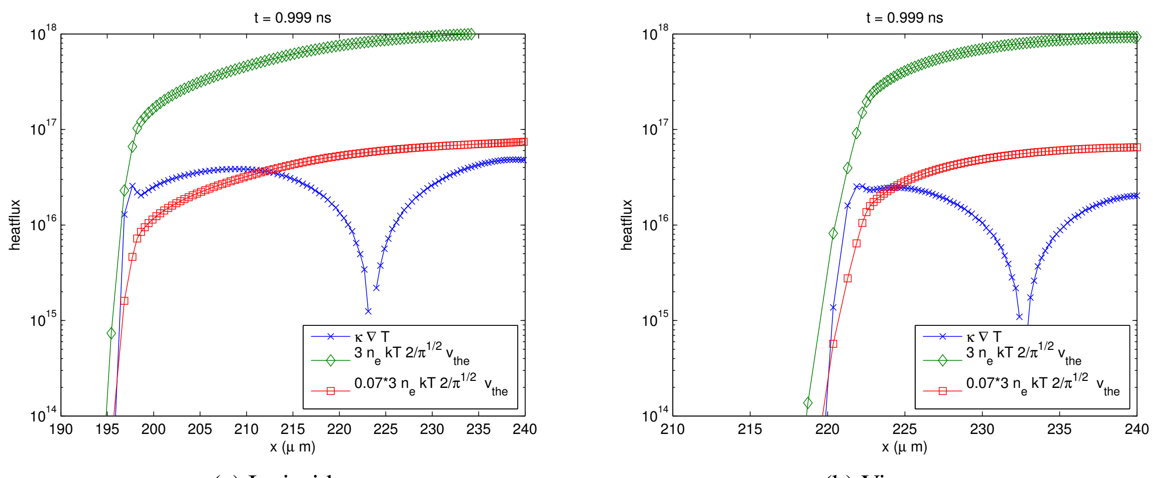


Figure 82: Neutron production rate over time for both simulations. The peak occurs due a the time of maximum compression as indicated in the previous plots.

The plot of the neutron production rate in fig. 82 also illustrates the difference in shock timings. The inviscid case lacks a sharp peak at the time of incidence of the first shock because the lack of viscosity does not allow for much compression at the time of first shock, this is somewhat evident from fig. 78. While the viscous shock decompresses significantly during the time between the first and reflected shock incidence, the inviscid case does not since it does not compress strongly. Along with the decrease in temperature indicated in fig. 81, this results in a flat profile for the neutron production rate from inviscid simulation during the time between the two shocks, while the viscous case undergoes a clear drop in production rate until the time of compression burn is reached. It should also be noted that the inviscid case has a higher maximum neutron production rate despite the fact that the maximum compression achieved in this case is smaller by a factor of 150%. This is because of the large temperature discrepancy throughout the domain, which is partially reflected through the discrepancy in the 2nd temperature maximum in fig. 81.



(a) Inviscid (b) Viscous

Figure 83: Plot of the heat flux, $\kappa_e \nabla T_e$ vs the physical flux limit, $3nk_bT_e \times v$, and the commonly implemented flux limit, $7\% \times 3nk_bT_e \times v$

Figure 83 shows a plot of the heat flux, $\kappa_e \nabla T_e$ along with the physical flux limit and the commonly implemented flux limiter of 7% of the physical flux limit [60, 42] at a time when an equal amount of energy has been deposited in both simulations and before the time of first shock. The quantity that is actually plotted is the magnitude of the heat flux, $|\kappa_e \nabla T_e|$, in comparison to the magnitude of the internal energy that can be moved by a characteristic velocity, $v = \frac{2}{\sqrt{\pi}} v_{th}$.

The comparison for the viscous and inviscid case illustrates a few things. Again, the discrepancy in the size of the shock is very evident. The inviscid flux spans $195 - 240$ μm while the viscous case is smaller by 25 μm. In the case of inviscid flux, a region of 15 μm spanning $195 - 210$ μm would require the implementation of a flux limiter. Since the shocked region is not as long due to viscous diffusion in fig. 83(b), the region that would require a flux limiter only spans 2 μm. This suggests that viscous effects help play a role in reducing the usage of a flux limiter. Since these flux limiters are implemented in an ad-hoc fashion, restricting their use allows to reproduce physics without artificial effects. In general, it can be said that the difference in magnitude of the heat flux with respect to the flux limiter is much greater in the viscous case.

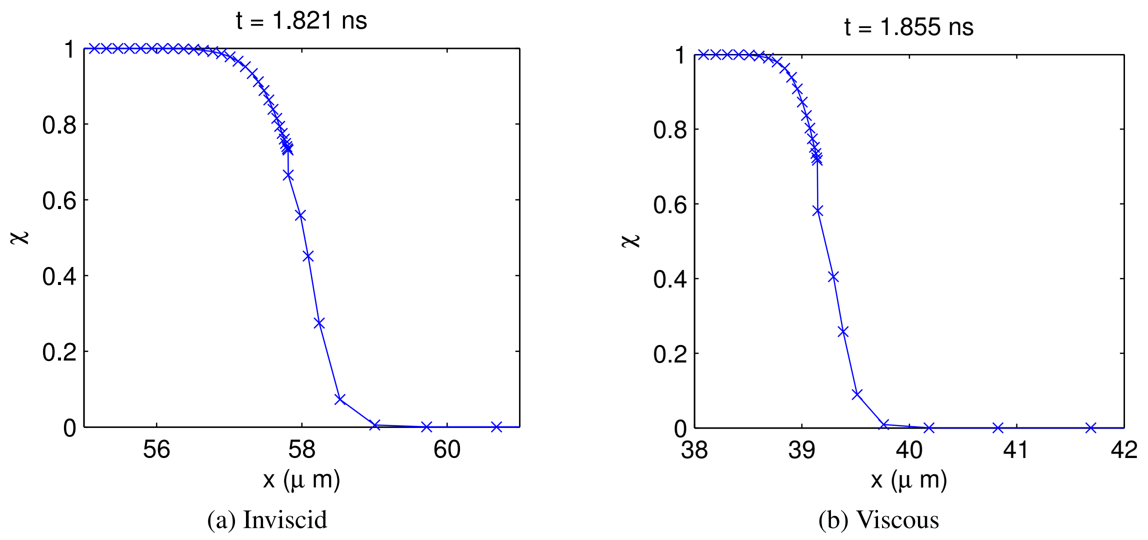


Figure 84: Plot of the mixing layer between DD and CH at the time of compression burn for (a) the Inviscid Simulation and (b) Viscous Simulation

Figure 84 shows the difference in the mass mixing interface at the time of maximum compression. As has been written, the location of the mixing layer is different by approximately 20 μm because the inviscid case does not replicate the compression of the viscous case. This is also indicated by the size of the mixing layer itself. Figure 84(a) shows that the mixing layer for th the inviscid case is approximately 2.5 μm in width while that of the viscous case is approximately smaller by a factor of 2. Post-processed calculations from a HELIOS simulation shows good agreement with the width in the viscous case.

Conclusion

In this study, the authors modified an existing 1 dimensional, 1 temperature, staggered grid Lagrangian Hydrodynamic to include three temperature effects for ions, electrons, and radiation. This was done in order to more accurately model DD-CH implosions such as those performed at the Omega Laser Facility. Along with adding three temperature effects, the code was also augmented with viscous effects such as viscous diffusion of momentum as well as viscous heating of ions as a result of the momentum diffusion.

These two effects helped reproduce experimental results as well as calculations from other codes to good agreement. The incidence of the first shock was approximately 1.55 ns into the simulation and the return shock arrived at 1.86 ns. The shock timings agree well with previous calculations and measurements from [64, 52, 15, 25]. The minimum radius of the fuel during compression reached 40 μm, not quite matching the HELIOS calculations of an $r_{\min} \approx 20\ \mu m$. The neutron production rate was approximately 10^{21} n/s which matched previous calculations and measurements to the order of magnitude. Additionally, concerns about achieving the correct burn weighted ion temperature of approximately 7 keV to match the results posted in [15] were assuaged as the maximum burn weighted ion temperature was near 7.2 keV at the time of the first shock.

One of the intentions of using this code was to examine the effect of viscosity on the ICF implosions. The previously mentioned results included viscous effects. Removing the influence of viscous diffusion and viscous heating of the ions resulted in differences of more than 10% in the shock timings, maximum fuel compression, neutron production rate, and burn weighted ion temperature. The shock arrives earlier by a difference of approximately 0.2 ns. The fuel compresses to a minimum of 55 μm, larger by 15 μm. It is evident from the results that viscous effects cause a decrease in the size of the shocked region. This decrease in the size of the shocked region is the reason behind the discrepancies in the shock timing. Since the shocked region is larger in the inviscid case, the shock arrives earlier. It also takes a longer time to reflect from the origin back to the plastic back to the origin because the shocked region has to transfer energy from the origin to the plastic region near 60 μm. Additionally, the ion temperatures are consistently higher for the inviscid case due to the lack of momentum diffusion, this results in approximately a factor of 2 difference in the ion temperatures, and consequently the burn weighted ion temperatures. Given that the burn weighted ion temperatures are much larger, it follows that the neutron production rate is also augmented in the inviscid case despite the relative lack in compression. The mixing layer widths calculated from post-processed HELIOS exhibit better agreement with the viscous case than the inviscid. The calculation from HELIOS resulted in a width of approximately 1.5 μm which the viscous came close to reproducing. The width for the inviscid case is larger by a factor of 2, approximately 2.5 μm. Finally, it was also determined that the implementation of viscous effects helped reduce the need for a electron heat flux limiter in these hydrodynamic simulations. Flux limiters are commonly used to avoid unphysical heat fluxes due to very steep temperature gradients. Since the viscous terms helped increase the size of the shock front as well as decrease the magnitude of the ion temperature in the shocked region, the need for a flux limiter only encompassed a few microns of the whole domain. This was in direct contrast to the inviscid case which had a larger, $\approx 20\mu m$, region that may require the implementation of a flux limiter.

It was determined that viscous effects play a crucial role in implosion dynamics for this DD-CH model and that all codes should consistently include viscous effects to model these implosions correctly. Viscosity strongly affects shock timings, fuel compression, neutron production rates, and heat fluxes in this case, and should do so in other direct drive ICF capsule modeling such as D^3He, DT, and others since all fuel choices are light and generally get hot.

New Methods for Profiling Shocks in Multidimensional Lagrange Calculations

By Brian O'Neill and Ben Magolan,
Alan Harrison and Scott Runnels (Mentors)

Abstract

A new method for profiling shocks in multi-dimensional Lagrange calculations has been devised and implemented into the LANL rad-hydro code `FLAG`. The model is an extension of the 1D formulation proposed by Nelson (2009) which uses a linear extrapolation of the gradient of displacement to identify the location of the shock. This 1D formulation has been generalized to 2D in the present model, which incorporates the use of a least squares planar fit of displacement with a series of filtering criteria to identify valid shock estimates. These estimates are used to reconstruct an estimated shock front through application of a least squares linear fit and interpolation scheme. This entire process has been fully implemented in `FLAG` and the resulting shock points and shock front can be rendered and visualized in *EnSight*.

Introduction

One of the ways shock waves in computational simulations of compressible hydrodynamics differ from real-world shocks is that while the width of a real-world shock is negligible compared to the physical scale of the system (the Rankine-Hugoniot jump conditions are derived from assuming a vanishingly small shock transition region), a numerical shock smears the transition over a few mesh zones typically, even though the jump conditions may still be faithfully reproduced far from the shock. The smearing results from mapping continuous variables onto a mesh and discretizing the partial differential equations that govern the system. The exact width of the numerical shock will depend on the resolution used and the method by which the compressible Navier-Stokes equations are being solved. Therefore, determining shock location as well as shock velocity is a non-trivial issue.

Frequently used methods for locating a shock in numerical simulations are based on analyzing material parameters (e.g. density, pressure) within and in the vicinity of the shock transition region. Typically, one finds the position at which some parameter in the flow has reached a specified fraction (e.g. $\frac{1}{2}$) of the total jump across the shock, or where the spatial gradient of a parameter reaches a maximum. Codes that employ artficial viscosity can use the location where it peaks. Interpolation can be used to estimate a sub-grid location, but such estimates are reliable only to a sizable fraction of the grid size. The estimated position is also subject to the choice of zones included in the stencil for either the averaging which determines the values of the material parameter in the pre- and post-shock regions, or those included in the interpolation. As the shock moves from one zone to another, zones are added to and dropped from the stencil which can cause discontinuities in time of the estimated shock position.

More precise data about the shock location and velocity in numerical hydrodynamics could be utilized by a variety of physics and engineering problems. The LANL rad-hydro code `FLAG` has packages that simulate the formation of ejecta particles as a Richtmyer-Meshkov instability, and the growth rate of this instability is a function of the speed of the shock passing through the material. In astrophysics simulations, the effect of diffusive shock acceleration, the process that energizes cosmic rays, is modelled as an injection of high energy particles into the simulation at the location of the shock front. There is also evidence that the efficiency of the acceleration process may be related to the strength of the shock, so precise shock position and velocity information could improve such models.

In light of the deficiencies of common practices for shock location, Nelson (2009) introduced the displacement method. In an ideal steady shock, the velocity of the fluid downstream of the shock is constant and thus the displacement of shocked fluid grows linearly in space and time. Therefore a plot of displacement versus position in the rest frame of the unshocked fluid would show a linear ramp in the downstream region which goes to zero at the shock position and is zero throughout the upstream region. Numerical smearing of the transition causes the steady shock to develop a curved foot, but the linear region of the ramp downstream can be used to extrapolate the shock location (see Figure 85).

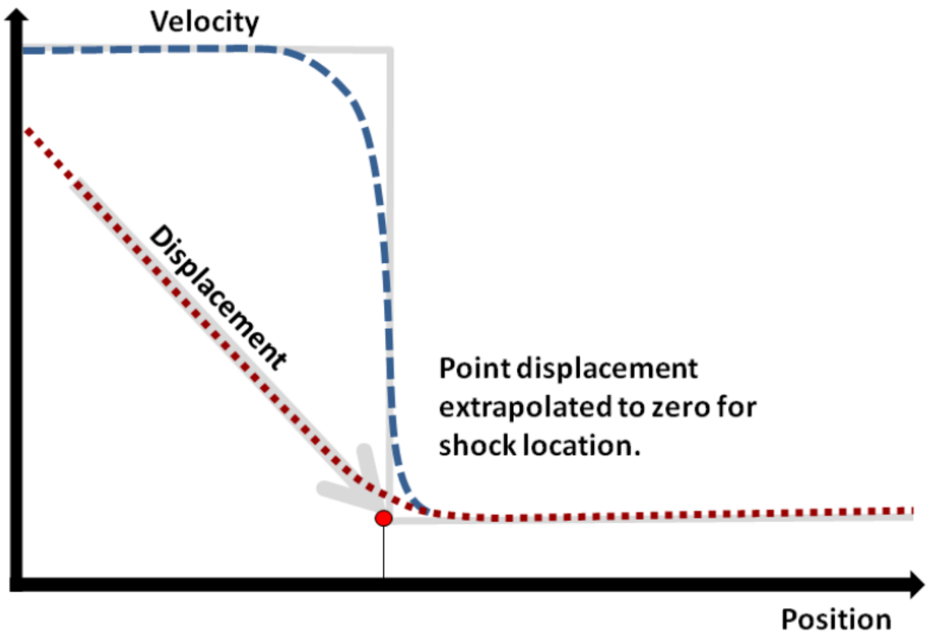


Figure 85: 1D displacement method, from Runnels & Margolin (2013)

In this work, we adapt Nelson's method to multiple dimensions by eliminating the need for sampling a large number of post-shock points. Doing so allows for the method to be applied to small patches of cells in a 2D or 3D simulation. We have incorporated this method into `FLAG` as a new module designed to be able to identify shock fronts in simulations of real-life engineering problems. We have also included routines to allow plotting of estimated shock fronts in the visualization software *EnSight*. We demonstrate this method for various common test problems as well as more complex cases. We discuss our results and future work that can improve the robustness of our method.

Method

Following Nelson, consider an ideal steady 2-dimensional shock propagating through space in the upstream rest frame, where the shock has velocity $\vec{u}_{\mathrm{sh}}$ and is locally planar and a downstream fluid element has velocity $\vec{u}_p \parallel \vec{u}_{\mathrm{sh}}$. An unshocked fluid element has position $\vec{x}_0$ until it is shocked at time t_{sh}, so the position of the fluid element for all time is

$$\vec{x}(t) = \vec{x}_0 + \vec{u}_p(t - t_{\mathrm{sh}})H(t - t_{\mathrm{sh}}), \tag{100}$$

where H is the Heaviside step function. For $t > t_{\mathrm{sh}}$, the location of the portion of the shock that encountered the fluid element is given by $\vec{x}_{\mathrm{sh}}(t) = \vec{x}_0 + \vec{u}_{\mathrm{sh}}(t - t_{\mathrm{sh}})$, and the difference in position between the fluid element and the shock is

$$\vec{d}(t) = \vec{x}_{\mathrm{sh}} - \vec{x} = (|\vec{u}_{\mathrm{sh}}| - |\vec{u}_{\mathrm{p}}|)(t - t_{\mathrm{sh}})\hat{u}_{\mathrm{p}}. \tag{101}$$

We define shock displacement as

$$\vec{x}_D = \vec{x} - \vec{x}_0 = \vec{u}_p(t - t_{\mathrm{sh}}) = \frac{\vec{d}}{|\vec{u}_{\mathrm{sh}}|/|\vec{u}_{\mathrm{p}}| - 1} \tag{102}$$

and since $\nabla|\vec{d}| = -\hat{d}$, the gradient of the magnitude of displacement is given by

$$\nabla|\vec{x}_D| = \frac{\hat{d}}{1 - |\vec{u}_{\mathrm{sh}}|/|\vec{u}_{\mathrm{p}}|}, \tag{103}$$

which is constant in space and time behind the shock and points in the opposite direction of $\vec{d}$, consistent with a displacement field that is zero at the shock position, and increases linearly in the downstream region.

Thus, if one knows the magnitude of displacement and its gradient at an instant of time somewhere in the post-shock flow, then the current position of the shock can be deduced, given by

$$\vec{x}_{\mathrm{sh}} = \vec{x} + \vec{d} = \vec{x} - \frac{|\vec{x}_D|}{(\nabla|\vec{x}_D|)^2}\nabla|\vec{x}_D|, \tag{104}$$

and the speed of the shock can be ascertained from the magnitude of the gradient, $|\nabla x_D|$, and the post-shock flow speed, u_p, by taking the magnitude of Equation 103.

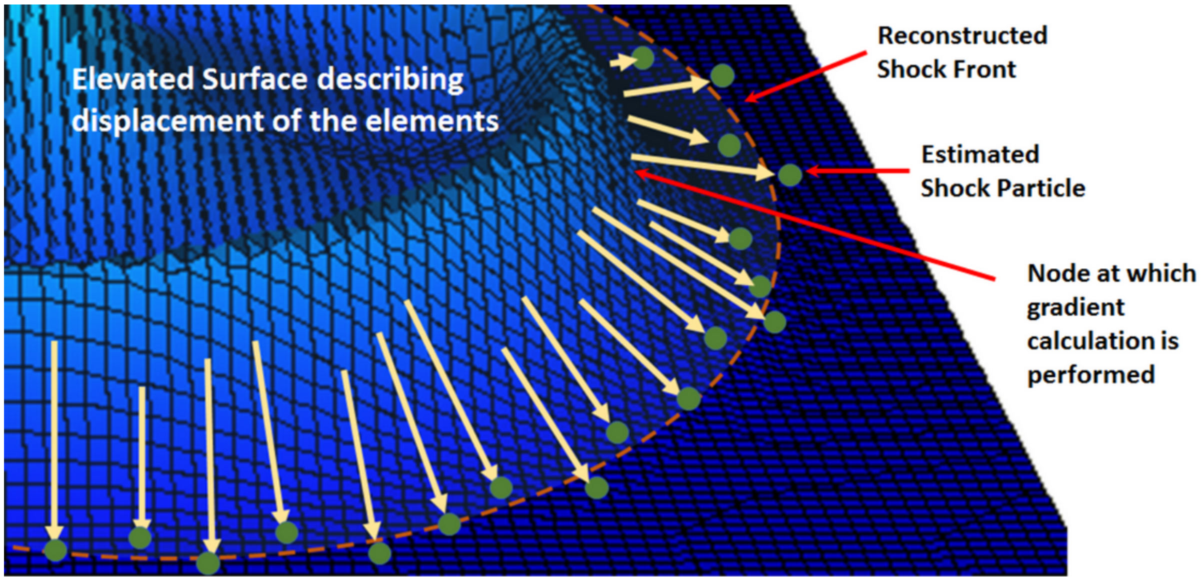

Elevated Surface describing displacement of the elements
Reconstructed Shock Front
Estimated Shock Particle
Node at which gradient calculation is performed

Figure 86: Cartoon of 2D method for identifying shock fronts

Nelson's method of performing linear regressions over large number of post-shocked points is, in principle, extendable to 2D. However, a difficulty arises in trying to determine along which line in the 2D mesh the points should be sampled. A simplification arises if, instead of considering a large number of post-shocked points to arrive at a single shock location, several combinations of just a few adjacent points are used to create several estimates of the shock location. Therefore, we look to identify shocked regions of the flow by performing least squares planar fits of the magnitude of displacement, x_D, at each mesh node, solving

$$\begin{bmatrix} \sum x_i^2 & \sum x_i y_i & \sum x_i \\ \sum x_i y_i & \sum y_i^2 & \sum y_i \\ \sum x_i & \sum y_i & \sum 1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} \sum x_i x_{D,i} \\ \sum y_i x_{D,i} \\ \sum x_{D,i} \end{bmatrix} \tag{105}$$

where the summations are over coordinates and values of x_D from the node's immediate neighbors. The solution provides the least squares plane $z = Ax + By + C$, so A and B approximate the x- and y-components of the gradient ∇x_D. The values of ∇x_D can be utilized to calculate a candidate shock position for each node on the mesh using Equation 104. Therefore, we are looking to utilize locally planar behavior of the displacement $x_D(\vec{x})$ to identify a "cloud" of candidate shock locations. These candidates can be accepted or rejected based on various criteria, the method for which we will elaborate more about in later sections. The accepted candidates should trace any shock fronts in the fluid. Fitting can be done to the positions of these accepted points to construct a shock front through the mesh. Our overall procedure is sketched out in Figure 86.

Validation

We validate our proposed method using a 1D piston problem. 100 zones distributed over $0 \leq x \leq 1$ filled with a γ-law gas ($\gamma = 5/3$) are compressed by a piston moving to the right with unit velocity. The initial density is $\rho = 1$, the specific internal energy is $e = 0.01$, and the simulation is evolved until $t = 0.7$. Profiles for the pressure and velocity are shown in Figure 87. The midpoint of the jump transition is between the centers of cells 92 and 93 according to the pressure profile, and between nodes 93 and 94 according to the velocity profile. Linear interpolation of the velocity and pressure data gives estimated shock positions of $x_{\mathrm{sh},u} = 0.9305$ and $x_{\mathrm{sh},P} = 0.9270$. The difference between these estimates and the analytical answer of $x_{\mathrm{sh}} = 0.9339$ is attributed to numerical error of the simple hydro code employed for this experiment.

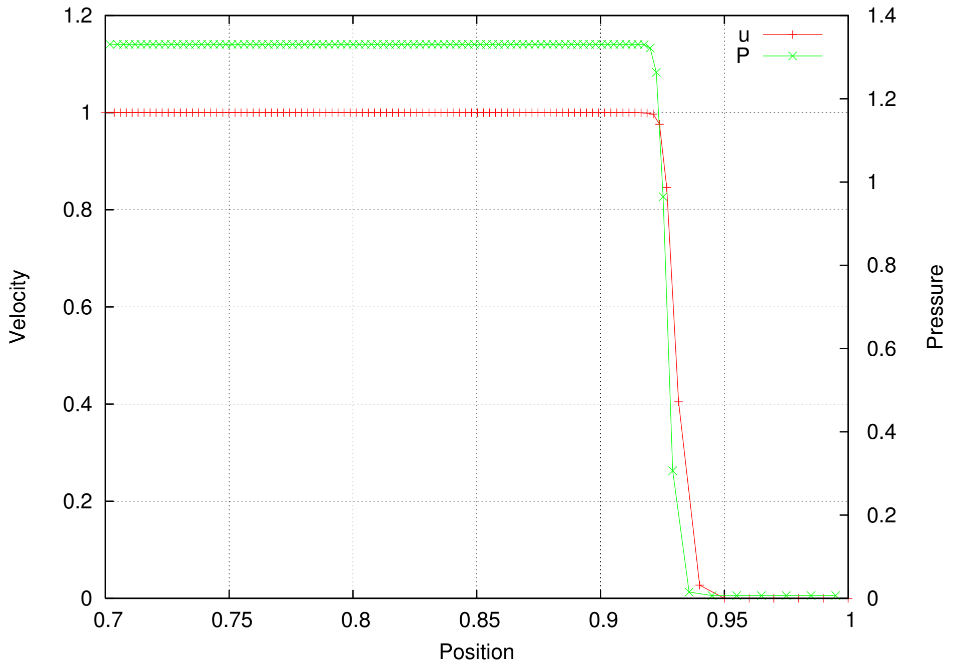


Figure 87: Velocity and pressure profiles from 1D piston

A least squares linear fit is performed to the values of displacement from nodes 50-89 and extrapolated to give an estimated shock position at $x_{\mathrm{sh},D_{40}} = 0.9282$. The goodness of fit can be tested by the sum of squared residuals (R^2), which, normalized to the square of average cell width in the fitting region, for our 40 node fit is $R^2 = 6.045 \times 10^{-7} \left(\overline{dx}\right)^2$. Least squares linear fits are also performed for each consecutive group of three nodes from 1 to 93 (91 fits total) and used to extrapolate potential shock positions. Figure 88 shows the majority of these estimates correlate with the more robust displacement fit indicated by the green line, and poorer estimates can be identified by higher values of R^2, here normalized by the square of the length of the smaller cell neighboring the central node of the fit, denoted $(dx)^2$.

If we reject all the points with $R^2 \geq (dx)^2 \times 10^{-6}$, our worst estimate of the shock position differs from $x_{\mathrm{sh},D_{40}}$ by 5.9×10^{-4}, only about 6% of the original mesh spacing. Most remaining estimates differ from $x_{\mathrm{sh},D_{40}}$ by less than 10^{-5}. Also, as the location at which the fit is being calculated moves closer to the shock position, the values for R^2 tend to decrease until the curved foot is reached. This means that if a shock is relatively steady, i.e., if the decay rate of the shock speed is small compared to the rate at which the shock passes through the cells of a numerical mesh, we should be able to utilize linear behavior of the displacement in localized regions

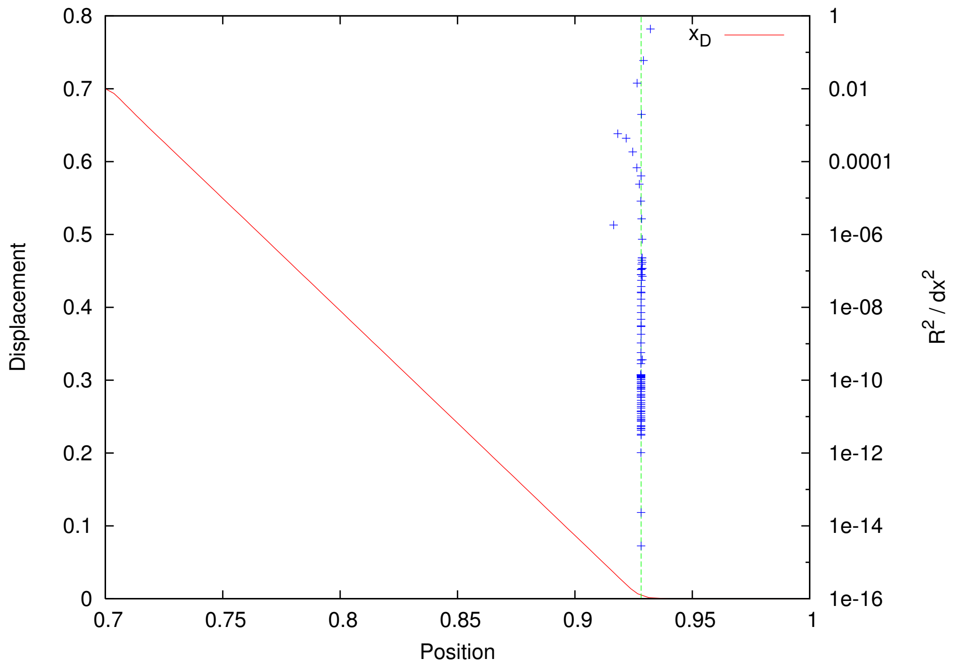


Figure 88: Shock estimates extrapolated from displacement method on 1D piston

of the post-shock flow immediately downstream of the curved foot to identify and locate the shock fronts in a simulation.

Filtration of Valid Estimated Shock Locations

Ideally, we would like to ultimately identify universal input settings and algorithms for identifying a variety of shocks. At present, our implementation requires manual adjustments to properly filter the good estimated shock locations for individual problems. In refining our method, we identified four overlapping ways to filter out poor estimates based on the results of our least squares planar fits. These filters are left dimensionless to maintain universal functionality, and are implemented as user-definable input settings to allow them to be adapted to different problems.

Goodness of fit

As mentioned before, R^2 gives a quantification of the planarity of the values included in our fits. If the R^2 value associated with the fit at a particular node is large, it is unlikely to provide an accurate shock location. Either the shock is too unsteady to allow for linear extrapolation, the node is located within the curved foot and we know the displacement there to be unphysical, or the flow has been otherwise altered (e.g. non-physical wall-heating). We define a characteristic mesh size associated with each node, Δx, which represents the distance to its nearest neighboring node. Our fitting tolerance, ε_{SSR}, is then made dimensionless by defining it as

$$\varepsilon_{\mathrm{SSR}} = R^2 / (\Delta x)^2 . \tag{106}$$

Magnitude of ∇x_D

The magnitude of the gradient, $|\nabla x_D|$, can be used as a flag for whether or not a node has experienced a shock. The displacement, x_D, in unshocked regions is locally constant in space, so $|\nabla x_D| = 0$. This magnitude is shown in Equation 103 to be a function of the ratio of the shock speed to the post-shock speed, $u_{\rm sh}/u_{\rm p}$, which itself is a function of the Mach number of the shock, $M_{\rm sh} = u_{\rm sh}/c_s$, where c_s is the speed of sound in the unshocked fluid. For a shock propagating through a gas following a γ-law equation of state, the magnitude of the gradient is therefore given by

$$|\nabla x_D| = \left[\frac{\gamma+1}{2}\left(\frac{1}{2}+\sqrt{\frac{1}{4}+\frac{M_{\rm sh}^2}{(M_{\rm sh}^2-1)^2}}\right)-1\right]^{-1}, \tag{107}$$

which approaches finite values of $|\nabla x_D| = 2/(\gamma-1)$ in the strong shock limit, and $|\nabla x_D| = 0$ in the weak shock limit. The gradient of displacement is already dimensionless, so we can define a minimum tolerance, $\varepsilon_{\rm mag}$, which will be the minimum value of $|\nabla x_D|$ we accept, limiting the sensitivity of our method to shocks stronger than

$$M_{\rm sh} \geq \sqrt{\frac{2+2\varepsilon_{\rm mag}}{2+\varepsilon_{\rm mag}(1-\gamma)}}. \tag{108}$$

Range of $|\vec{d}|$

We can attempt to isolate the "best" region for linear extrapolation by defining an acceptable range for $|\vec{d}|$, the value of which is seen by Equation 104 to be given by

$$|\vec{d}| = \frac{x_D}{|\nabla x_D|}. \tag{109}$$

The length of the curved foot sets a minimum scale for $|\vec{d}|$. The exact length of this foot depends on the width and shape of the shock transition, which will be discussed in more detail in the following section. The maximum scale of $|\vec{d}|$ can be set by a variety of problem dependent circumstances, e.g., the length scale over which an unsteady shock has appreciably decayed. A dimensionless range of acceptable $|\vec{d}|$ values can be set through normalizing by the local characteristic mesh size, Δx, defining $d_{\rm min}$ and $d_{\rm max}$ such that

$$d_{\rm min} \leq \frac{|\vec{d}|}{\Delta x} \leq d_{\rm max}. \tag{110}$$

Direction of ∇x_D

Other flow patterns, namely rarefactions, can display linear behavior of x_D in space, as seen in the profile of a 1D Sod problem in Figure 89. The initial setup for the Sod problem has two fluids in contact and at rest, region 1 at higher pressure and density ($P = 1, \rho = 1$) and region 2 at lower pressure and density ($P = 0.1, \rho = 0.125$). The fluid in both regions obeys a γ-law equation of state with $\gamma = 5/3$. The initial numerical setup has 400 zones spanning from

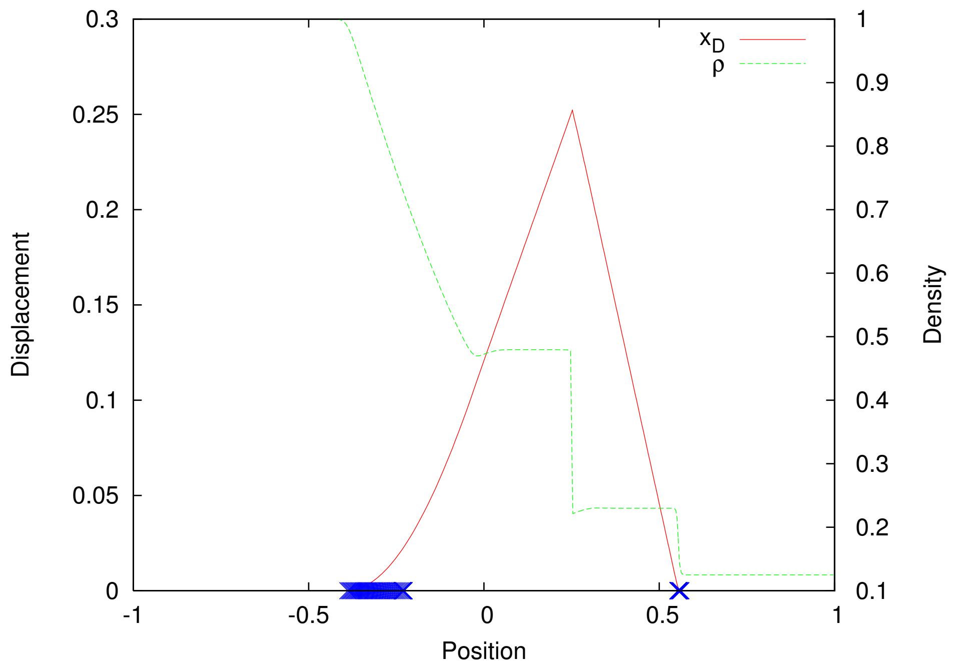


Figure 89: Profiles of displacement and density for 1D Sod shock tube problem, with localized shock position candidates filtered based on $\varepsilon_{\mathrm{SSR}}$ and $\varepsilon_{\mathrm{mag}}$

$-1 \leq x \leq 1$, units are cgμs. As the simulation evolves, a rarefaction wave propagates into region 1, while a shock wave and a contact discontinuity propagate into region 2. The Figure shows the problem evolved to $t = 0.3$, and while a linear ramp is properly identifying the current position of the shock to the right, the rarefied gas to the left of the contact discontinuity displays a similar displacement ramp feature which is extrapolated to a range of points in the neighborhood of the current position of the rarefaction characteristic.

In the rest frame of the undisturbed gas at the left and right ends of the domain, it is easy to identify candidates associated with the rarefaction portion of the simulation. All fluid in the domain that is in motion has a velocity towards the right. Candidates associated with the rarefaction wave were extrapolated from regions with a positive slope of the displacement ramp, while those associated with the shock wave were calculated based on a negative slope of the displacement. Generalizing to vector notation where $\vec{u}$ is the velocity of the fluid element at which the extrapolation is being performed, another filter can be defined based on the sign of $\vec{u} \cdot \nabla x_D$ in the rest frame of the undisturbed fluid. Namely, if $\vec{u} \cdot \nabla x_D > 0$, the extrapolation characterizes a rarefaction, while $\vec{u} \cdot \nabla x_D < 0$ is consistent with a shock. We acknowledge that this particular filter is not Galilean invariant, but works only in one specific reference frame. For this work, we have chosen to focus on shocks propagating through quiescent fluids and left the problem of generalizing our method to fluids not initially at rest for future work. A general method for identifying rarefactions propagating through fluid in motion would still rely on the direction of ∇x_D, but with respect to the direction of some velocity difference which characterizes local changes in the velocity field that would be indicative of a rarefaction.

Shock Structure

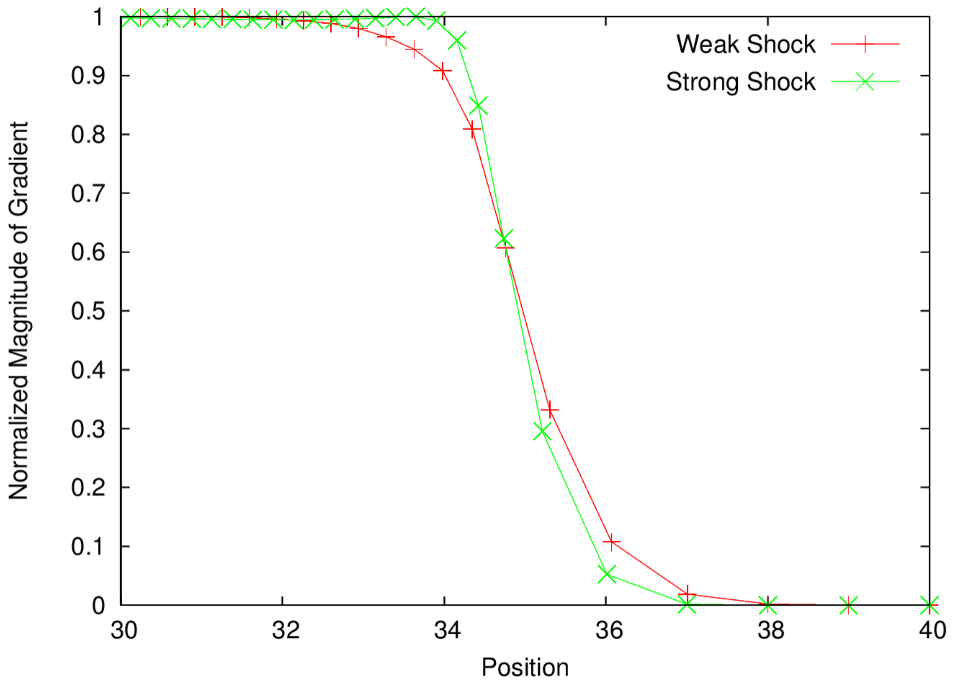


Figure 90: Profile of $|\nabla x_D|$ for 2D piston problems, demonstrating structure of strong and weak shocks

Through the course of our investigations, we discovered that the ideal filtering parameters for a problem are dependent on the particular shape of the numerical shock transition, which in turn can depend on the strength of the shock and the shock-capturing method employed, for example the values of artificial viscosity parameters used (Guenther et al. 1994). In the majority of our simulations, we used the `QBarton` implementation of artificial viscosity in `FLAG` (Runnels & Margolin, 2013), with various combinations of values for the q_1 and q_2 parameters. Figure 90 demonstrates the dependence of the width of the shock transition on shock strength with results from 2D piston simulations. For these simulations, a 40×40 mesh is used with initial node separation $\Delta x = 1$ cm. The shock position is initialized at $x = 5$ cm, the unshocked fluid has $\rho = 10^{-6}$ and $P = 10^4$ in cgs units, with the Rankine-Hugoniot relations used to initialize the state of the fluid in the post-shock region based on an analytical shock strength, M_{sh}, which results from the motion of a piston with Mach number M_{p}, the value of which defines the fixed velocity of the left boundary. The fluid everywhere obeys a γ-law equation of state with $\gamma = 5/3$. Artificial viscosity parameters of $q_1 = 1$ and $q_2 = 1$ were utilized in both cases. The weak case here is formed by a $M_{\mathrm{p}} = 2$ piston and the strong case by a $M_{\mathrm{p}} = 100$ piston. The simulation is evolved until the analytical position of the resulting shock has reached $x = 35$ cm. The Figure shows a slice along x of the profile of $|\nabla x_D|$ for each run. The strong shock case has effectively completed the shock transition and reached a constant post-shock value for $|\nabla x_D|$ within about 6 mesh zones. The weak shock, however, requires several more mesh zones to reach a constant value of $|\nabla x_D|$.

The displacement ramp can be thought of as the integral from right to left of the plots in

Figure 90. It is obvious then that the curved foot of the displacement ramp for the weak shock case trails several mesh zones farther back from the shock than in the strong shock case, i.e. the weak shock transition is wider. The worry here is that displacement values in the curved foot of a weak shock may be linear enough to pass our ε_{SSR} filter, but will be non-linear enough to provide a poor estimated shock position. We would then wish to filter out such candidates using our d_{min} parameter. Looking forward though, we can imagine utilizing our method on a simulation with multiple shocks at different strengths and may need different values of d_{min} to accurately trace the different shocks. Ideally, we would like to implement our filters to automatically calibrate themselves in response to the strength of a shock and the values of q_1 and q_2. We discuss these efforts in more detail later.

Shock Point Creation and Shock Front Reconstruction

After valid shock point estimates have been identified through use of the least squares gradient calculation in conjunction with the filtering criteria, this information is converted into a form that can be rendered by the graphics visualization package *EnSight*. The `FLAG` code has been modifed to support the plotting of individual shock points in addition to a reconstructed shock interface. This section chronicles the process associated with plotting these shock points and reconstructing the accompanying shock interface using the `FLAG` code architecture.

Shock Point Creation

As discussed in the previous section, the displacement gradient calculation coupled with the filtering criteria produces a "cloud" of valid shock estimates. These shock estimates are then converted into shock point data "particles" associated with the coordinates of the shock estimates that can then be plotted and rendered in *EnSight*. Note that while these shock points piggyback off of the "BParticle" class architecture, they are not to be considered particles in the traditional sense (such as with ejecta particles or tracer particles). Rather, the particle class architecture is used as a mechanism by which to visualize the shock point estimates.

The shock point "particles" also store additional information which can be quite useful when visualizing the data in *EnSight*. Variables that are stored with these shock points include:

- `shock_ssr`: R^2 value associated with least squares gradient calculation
- `shock_origin`: vector which points to the node from which the shock point estimate was created from
- `displace_grad`: vector characterizing magnitude and direction of least squares gradient calculation

These variable types can offer key insights regarding the origin of shock points and the nature of the shock wave. They can also be used as a debugging tool for identifying and selecting the appropriate screening criteria in the input file (e.g. R^2 and displacement gradient thresholds, characteristic lengths, etc.).

Shock Front Reconstruction

Reconstructing the shock interface requires the collection of shock points in a given zone's stencil, whereby a stencil is defined as all neighboring zones that share a node with a given zone of interest. Figure 91 depicts the construction of a standard stencil. If a sufficient number of shock points are identified in a zone's stencil, then a least squares linear fit is performed to identify the line of best fit that characterizes this data. Next, a linear interpolation scheme is utilized to determine if and where this line intersects the zone of interest. If there is an intersection, the resulting line segment is rendered in *EnSight* as the reconstructed shock interface for that zone.

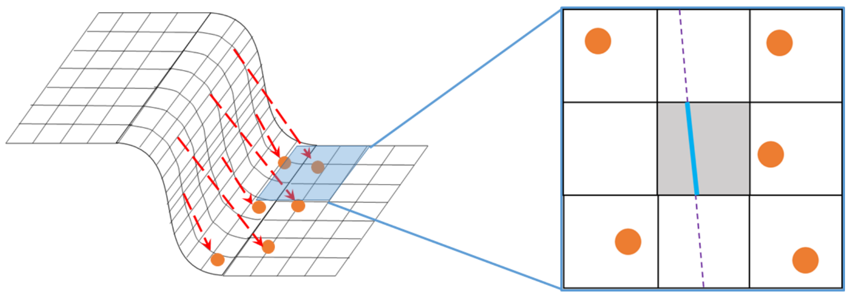


Figure 91: Stencil for LS Linear Fit

It is worth highlighting that the structured Cartesian stencil and mesh displayed in Figure 91 is used for clarity of presentation. The routine used to reconstruct the shock interface is completely general and works for fully unstructured meshes.

Least Squares Calculation

As discussed above, if a sufficient number of shock point "particles" are collected in a zone's stencil, then a least squares linear fit is performed on the shock point coordinates in order to identify the line of best fit (of the form $y = Ax + B$). The following equation is used to determine the values of the slope, A, and intercept, B, for a given zone's stencil:

$$\begin{bmatrix} \sum_{i=1}^{m} x_i^2 & \sum_{i=1}^{m} x_i \\ \sum_{i=1}^{m} x_i & \sum_{i=1}^{m} 1 \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{m} x_i y_i \\ \sum_{i=1}^{m} y_i \end{bmatrix} \tag{111}$$

where x_i and y_i denote the 2D coordinates of a given shock particle. The quantities A and B can be solved by computing the determinant and inverting the 2x2 matrix:

$$\begin{bmatrix} A \\ B \end{bmatrix} = \frac{1}{(\sum_{i=1}^{m} x_i^2)(\sum_{i=1}^{m} 1) - (\sum_{i=1}^{m} x_i)^2} \begin{bmatrix} \sum_{i=1}^{m} 1 & -\sum_{i=1}^{m} x_i \\ -\sum_{i=1}^{m} x_i & \sum_{i=1}^{m} x_i^2 \end{bmatrix} \begin{bmatrix} \sum_{i=1}^{m} x_i y_i \\ \sum_{i=1}^{m} y_i \end{bmatrix} \tag{112}$$

A caveat to this formulation exists for near vertical lines, where minimizing the y residuals fails. For this instance, the range of x- and y-values are examined to determine which distribution is tighter, and therefore should be the dependent variable for the linear fit. If the spread of x-values is tighter than that of the y-values, then an equation of the form $x = Ay + B$ is solved using a least squares linear fit for A and B that has a similar formulation as in Equation (112).

Interface Reconstruction Calculations

With the linear fit identified and the values for A and B obtained, it is then possible to calculate the vertices for the line segments which will be plotted in a zone of the mesh to portray the shock front. The shock front reconstruction maps a bar line segment through the zone. Therefore, the vertices of these bar segments will also be on the boundary of the zone.

Consider points p_1 and p_2 corresponding to an edge of a given zone, with coordinates (x_1, y_1) and (x_2, y_2), respectively. Simple linear interpolation can be used to identify the coordinates (x, y) of the intersection of the edge with the least squares line described by $y = Ax + B$. The equation is:

$$\frac{y_2 - y_1}{x_2 - x_1} = \frac{y_2 - y}{x_2 - x} \tag{113}$$

Substituting in $y = Ax + B$ yields:

$$\frac{y_2 - y_1}{x_2 - x_1} = \frac{y_2 - Ax - B}{x_2 - x} \tag{114}$$

Performing a bit of algebra produces the following expression for x:

$$x = \frac{(y_2 - y_1)x_2 - (x_2 - x_1)(y_2 - B)}{(y_2 - y_1) - A(x_2 - x_1)} \tag{115}$$

The y-coordinate can then be determined by plugging the value of x back into the equation $y = Ax + B$.

Generating both coordinates for the bar segment shock interface necessitates looping around all the sides of the zone and performing the linear interpolation calculations outlined above. There are three cases which describe the interaction of the least squares fit line with a given zone and side, as chronicled in Figure 92:

- The line $y = Ax + B$ intersects the given zone and side of interest
- The line $y = Ax + B$ intersects the given zone, but does not intersect the side of interest
- The line $y = Ax + B$ does not intersect the zone of interest

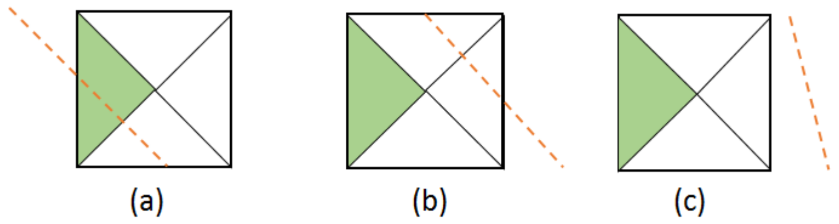


Figure 92: Possible Interactions of LS Fit with Zone of Interest

When walking around the sides of a zone, the coordinates for the potential interface are calculated using the linear interpolation scheme. These coordinates are then examined to determine whether or not they lie on an edge of the zone. Figure 93 illustrates this process. As can be seen in Figure 93(a), if the line travels through the side of interest, then the coordinates produced by the linear interpolation scheme will line up on the corresponding edge. However, if the line does not traverse the side, then the coordinates produced by the linear interpolation scheme will show up at the intersection of the extrapolation of the side's edge and the linear fit, which is clearly outside the zone as demonstrated in Figure 93(b).

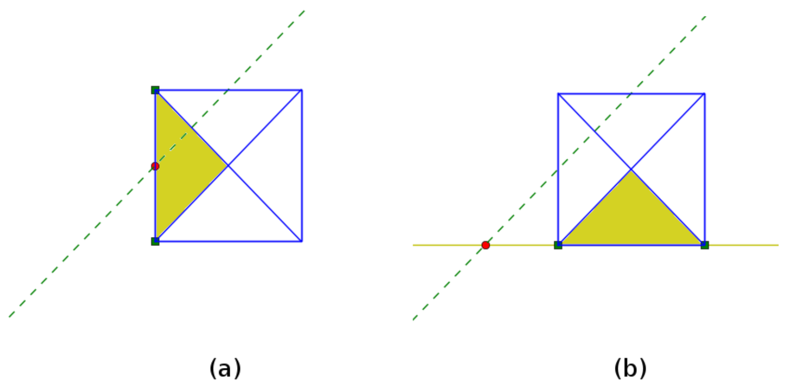


Figure 93: Screening and Identification of Interface Coordinates

It is worth noting that a uniform structured mesh diagram is used for the sake of clarity. The method has been implemented for fully unstructured meshes and should therefore work for general polygonal zones

Results

A series of simulations were performed in order to evaluate the utility of the shock profiling model outlined above. First, the general features and capabilities of the model are demonstrated through the use of several simulations. Next, the model's limitations with respect to the sensitivity of the filtering criteria are explored. The section concludes with an examination of the model applied to a challenging test case designed to probe the model's applicablity to more realistic engineering problems.

General Features of the Model

The shock profiling model applied to a weak Sedov problem on a 50x50 2D Cartesian mesh is shown in Figure 94. The shock wave emanates from the bottom left cell and the mesh is colored by pressure. This simulation was run on 4 processors in order to test the parallel implementation for communication across processor boundaries on domain decomposed meshes. At the front of the shock wave, there is an even distribution of shock points and the reconstructed shock interface spans almost the entire wave.

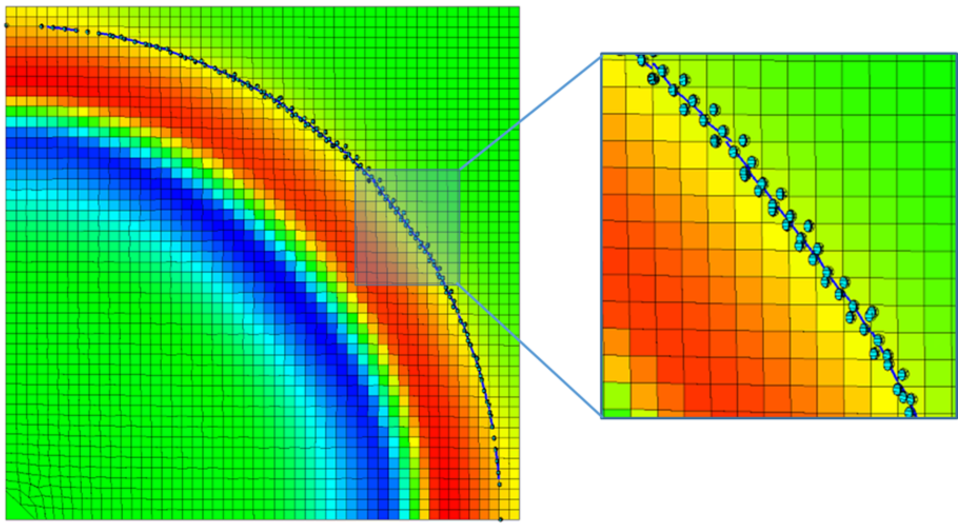


Figure 94: Reconstructed Shock Points and Shock Front for Weak Sedov Problem

The model has also been applied to a piston problem and an angled Sod problem, as illustrated in Figure 95. As with the weak Sedov problem, the meshes are colored by pressure. For the piston problem, the fluid is compressed from left to right by a piston with Mach number $M_{\mathrm{p}} = 10$ on a 40x40 2D Cartesian mesh. Only 1 processor was used for this simulation. For the Sod problem, the materials are partitioned along the diagonal at the start of the simulation. The fluid in the left region has a pressure and density that are precisely 10 and 8 times greater than their respective counterpart parameters in the region on the right. The simulation was performed on 50x50 2D Cartesian mesh in parallel using 4 processors, assuming a perfect gamma law constant of 5/3.

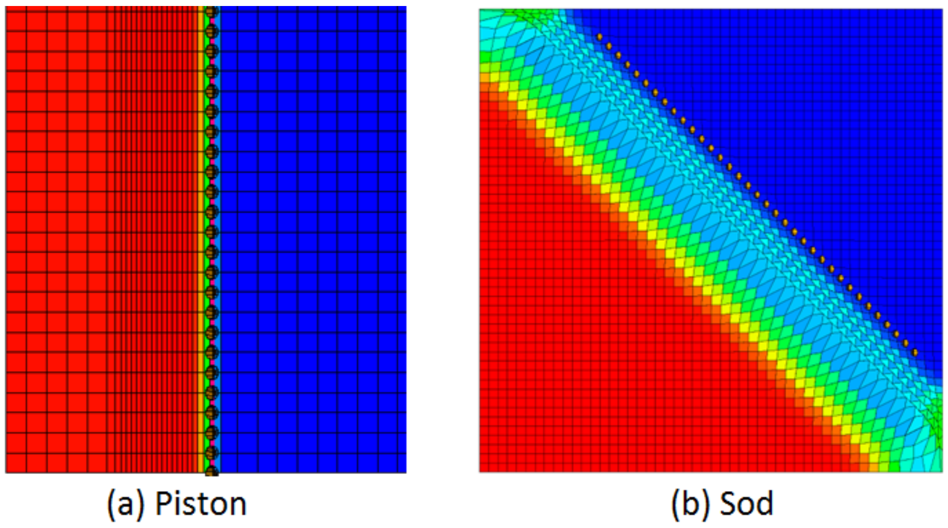


Figure 95: Additional Test Cases

The special features that the shock point data structures contain are demonstrated for a stronger Sedov problem in Figure 96. Figure 96(a) portrays vector arrows ranging from the shock point data structure to the node from which that shock estimate was produced. Figure 96(b) shows these same shock point data structures colored by their respective SSR (sum of squared residuals) values. These features are very useful for gaining insight into the nature of the shock wave and for trying to debug an input file to identify and select the appropriate filtering criteria by which to frame the simulation. For example, the vector arrows feature can be used as a visual check to identify where the shock points derive from and the SSR coloring scheme can be used to assess the "strength" of the shock estimate. Together, these two sources of information can be used to select a new SSR screening criterion in addition to modifying the minimum and maximum characteristic length tunable filters.

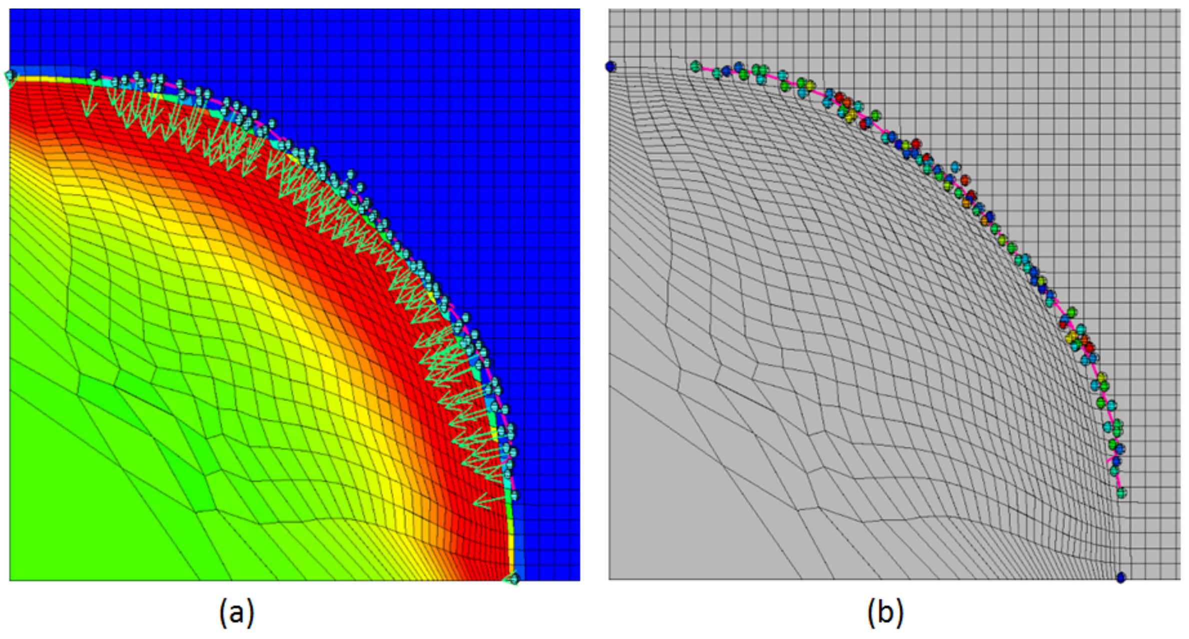


Figure 96: Special Features of the Shock Point Data Structures

Sensitivity of the Filtering Criteria

The sensitivity of the tunable filters can be explored by returning to the weak Sedov problem that was presented in Figure 94. Snapshots of the pressure contours along with the rendered shock points and reconstructed shock fronts for three distinct timesteps are shown sequentially in Figure 97. Near the beginning of the simulation there are few shock points and shock fronts visualized. As the simulation progresses, more and more shock points begin to appear. Towards the end of the simulation, it is observed that almost the entire shock front has been mapped with shock points and the shock interface.

The abundance of shock points at the end of the simulation versus the lack of points in the beginning is due to the selection of tunable filters for this problem. If the filtering criteria were to be relaxed (e.g. by increasing the R^2 tolerance, reducing the magnitude of displacement gradient threshold, and/or increasing the acceptable range of $|\vec{d}|$), then a respectable shock front can be imaged at earlier timesteps, as indicated in Figure 98(a). However, as the simulation

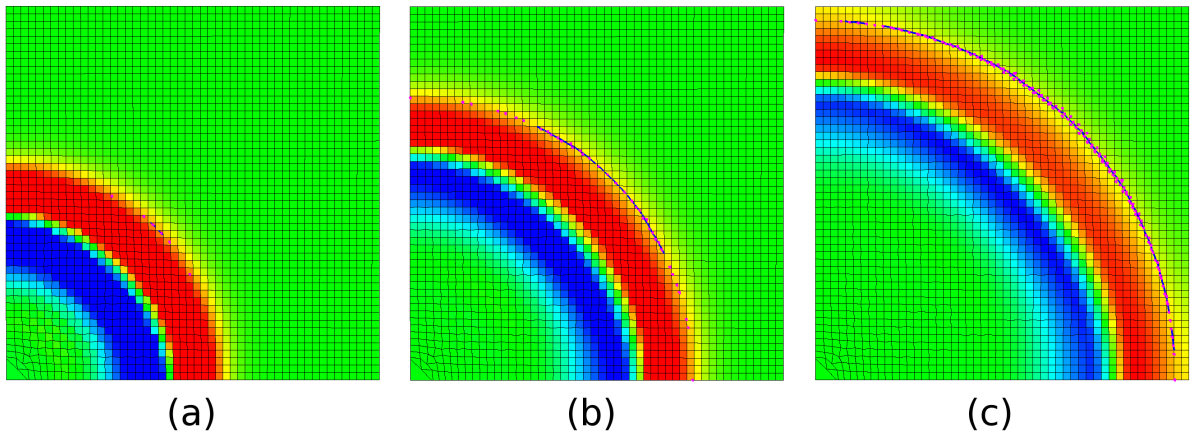


Figure 97: Development of Shock Points and Shock Interface for Weak Sedov Problem

progresses, it is plagued by spurious overproduction of shock points and shock fronts resulting in a convoluted shock interface, as shown in Figures 98(b) and (c). The sensitivity of the tunable filters on the production of shock points is a driving motivation for future work on the implementation of universal, scalable filtering criteria.

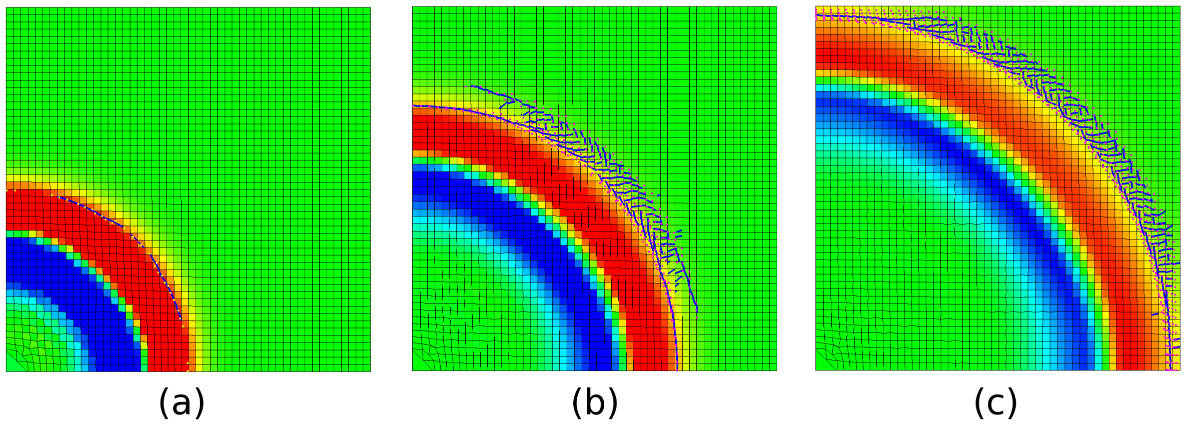


Figure 98: Development of Shock Points and Shock Interface for Weak Sedov Problem with Relaxed Filters

Extension to More Realistic Engineering Problems

In order to extend the model into the realm of realistic engineering applications, the weak Sedov problem portrayed in Figure 94 has been modified to incorporate the presence of obstacles in the direct path of the shock wave, as indicated by the white blocks overlaying the mesh in Figure 99. The simulation was performed on a 90x90 2D Cartesian mesh using 9 processors. Again, the shock wave emanates from the bottom left cell. The shock wave interaction with the obstacles gives rise to the unique pressure fields highlighted in Figures 99(a) and (b). Unfortunately, the shock points and shock front do not begin to develop until later in the simulation.

Even so, the many gaps in the shock interface are readily apparent. Both the utility and potential of the shock profiling model must not go unnoticed, however. Identifying the location of the shock front with only aid of the human eye is quite difficult for a problem of this nature where the pressure field is quite complex. Further refinement of the described model will hope to better capture these shock fronts, thereby adding clarity and insight to a challenging problem.

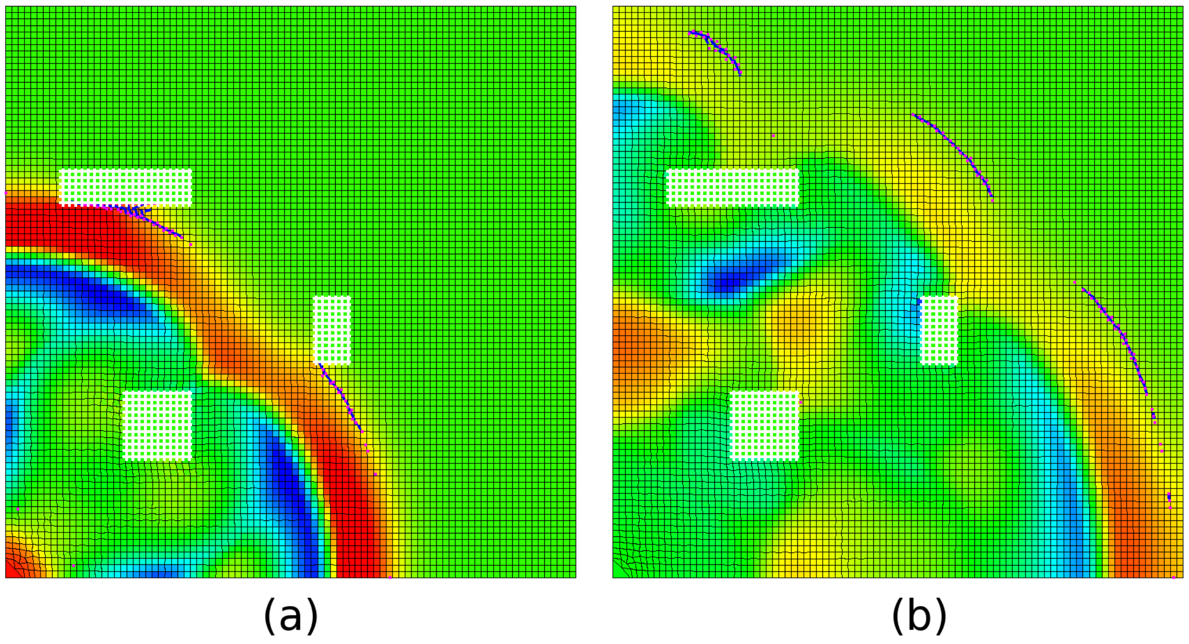


Figure 99: Development of Shock Points and Shock Interface for Weak Sedov Problem with Obstacles

Future Directions

We have demonstrated the utility of our current implementation for accurately estimating the position of 2D shocks in Lagrangian hydrodynamics simulations. We are able to trace shock fronts of various strengths and geometries, not only for idealized test cases, but for more complex problems as well. Still, there is plenty of work that can be done to further generalize and improve our method, and allow users to apply our shock location algorithms "out of the box" without the need for fine tuning to each particular problem.

Generalizing to Non-quiescent Flows

A big deficiency of our current implementation is that we are confined to working with shocks propagating through initially quiescent fluids. This is because the value for displacement, x_D, in our code is defined as the distance a node has moved since the beginning of the simulation, $x_D = |\vec{x}(t) - \vec{x}(t_0)|$. A proper definition of shock-induced displacement needs to account for initial velocity and should represent the distance between where a node is and where it would have been had it not encountered a shock, explicitly given by

$$x_D = |\vec{x}(t) - (\vec{x}(t_0) + \vec{u}_0(t - t_0))| \tag{116}$$

We may want to allow for the possibility that multiple shocks could encounter the same node over the course of a run, in which case we may require multiple displacement values for a node, $x_{d,i}$, representing the displacement caused by shock 1, shock 2, etc.

Alternatively, we could keep our current definition of displacement and take advantage of the local constancy of x_D in unshocked regions at an instant of time and instead of using $|\nabla x_D|$ to extrapolate to $x_D = 0$, determine a value $x_D(\vec{x},t)$ that represents the local baseline of the unshocked flow and extrapolate to that value for estimating the shock position. As previously stated, another method for distinguishing rarefaction from shocks will be required for non-quiescent problems.

Higher Order Fitting

The linearity of x_D is dependent on a constant shock speed, u_{sh}. If u_{sh} decays, the velocity field is not constant behind the shock and a linear fit to displacement is no longer appropriate, as seen in Figure 13 from Nelson (2009). We have employed our 1st order method on unsteady shocks such as those found in the Sedov problem, and reconstructed reasonably accurate shock fronts utilizing the quasi-linear behavior in the displacement ramp of the Sedov problem. We have also seen how our method requires a higher value of ε_{SSR} for identifying the shocks earlier in the simulation as opposed to at later times, which is to be expected from the analytical function defining the position of the Sedov shock front, given by

$$r_{\text{sh}} = \frac{1}{Q}\left(\frac{E_0 t^2}{\rho_0}\right)^{1/5} \tag{117}$$

where Q is a numerical factor from integrating the self-similar solution, E_0 is the energy deposited into the blast, and ρ_0 is the initial density. The shock speed varies as $u_{\text{sh}} \sim t^{-3/5}$ and the deceleration of the shock varies as $a_{\text{sh}} \sim t^{-8/5}$, meaning the decay rate of the shock speed varies as $\eta = a_{\text{sh}}/u_{\text{sh}} \sim t^{-1}$. Thus at large times the decay rate is smaller and the behavior of x_D is more linear.

Nelson demonstrated how fitting higher order polynomials to x_D can be used for unsteady shock transitions like the Sedov problem. Quadratic, cubic, and quartic fits to post-shock data are able to more precisely identify the shock position, utilizing data over a larger stencil. We may wish to include the ability to fit post-shock displacement data to higher order paraboloid or cubic surfaces to better characterize x_D. Such calculations are much more computationally intensive, and this cost must be taken into account in potentially adapting higher order methods.

Utilizing Time Dependence of ∇x_D

As previously mentioned, the efficacy of our implementation is dependent on understanding the shape of the shock transition. Currently, properly filtering candidate shock positions requires fine-tuning of the filtering tolerances. This is because, as Guenther et al. (1994) demonstrate, the width of the shock transition in a Lagrangian simulation employing artificial viscosity is dependent on the strength of the shock, M_{sh}, and the artificial viscosity parameters, q_1 (linear coefficient) and q_2 (quadratic coefficient). Most noticeably for our purpose, the length of the curved foot of a shock front's displacement ramp is longer for weaker shocks and for higher values of q_1, which may influence our choice for the range of $|\vec{d}|$.

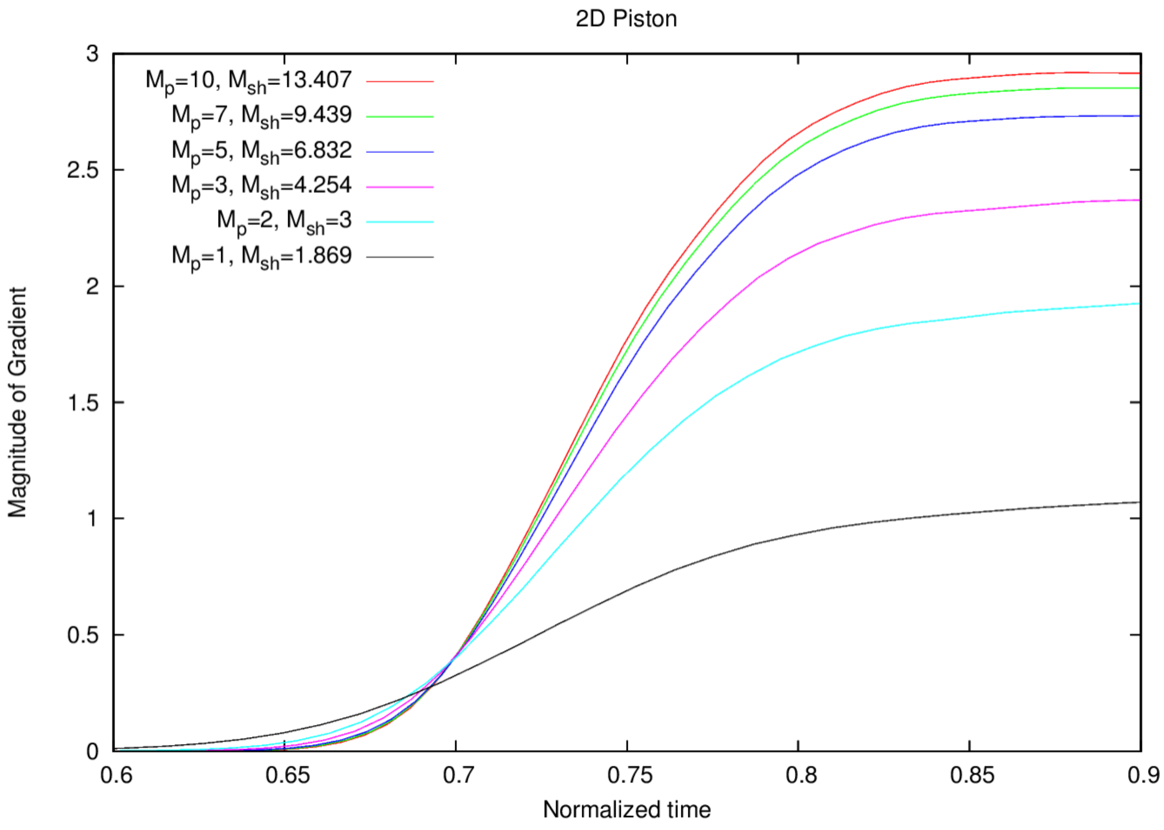


Figure 100: $|\nabla x_D|(t)$ as measured at a particular node in the mesh of a 2D piston problem for a range of shock strengths, demonstrating shape of shock transition. Different runs identified by the Mach number of the piston, M_{p}, and the analytical Mach number of the resulting shock,M_{sh}. Artificial visosity is implemented with $q_1 = 1$ and $q_2 = 1$. Time values normalized to end time of each simulation.

Figures 100 and 101 demonstrate the similar behavior of $|\nabla x_D|$ as measured from a single node as the shock front passes through. Each of these curves is similar in the fact that they all have some time t_{infl} at which the time derivative of $|\nabla x_D|$ is a maximum, i.e. an inflection point. Furthermore, the value of $|\nabla x_D|$ at $t = t_{\mathrm{infl}}$ is roughly half the value ultimately reached once the shock transition is complete. We propose there exists some function of shock strength and artifical viscosity parameters which can relate the post-shock value of $|\nabla x_D|$ to its value at the inflection point, or explicitly

$$|\nabla x_D|_{t\to\infty} = C(M_{sh}, q_1, q_2, \ldots)|\nabla x_D|_{t=t_{\mathrm{infl}}}. \tag{118}$$

Thus, if one tracks $\frac{d}{dt}|\nabla x_D|$ in order to identify $|\nabla x_D|_{t=t_{\mathrm{infl}}}$, one could estimate the strength of the shock via Equation 107, which could be used to properly configure the range of $|\vec{d}|$ that would supply the most accurate estimates of shock position.

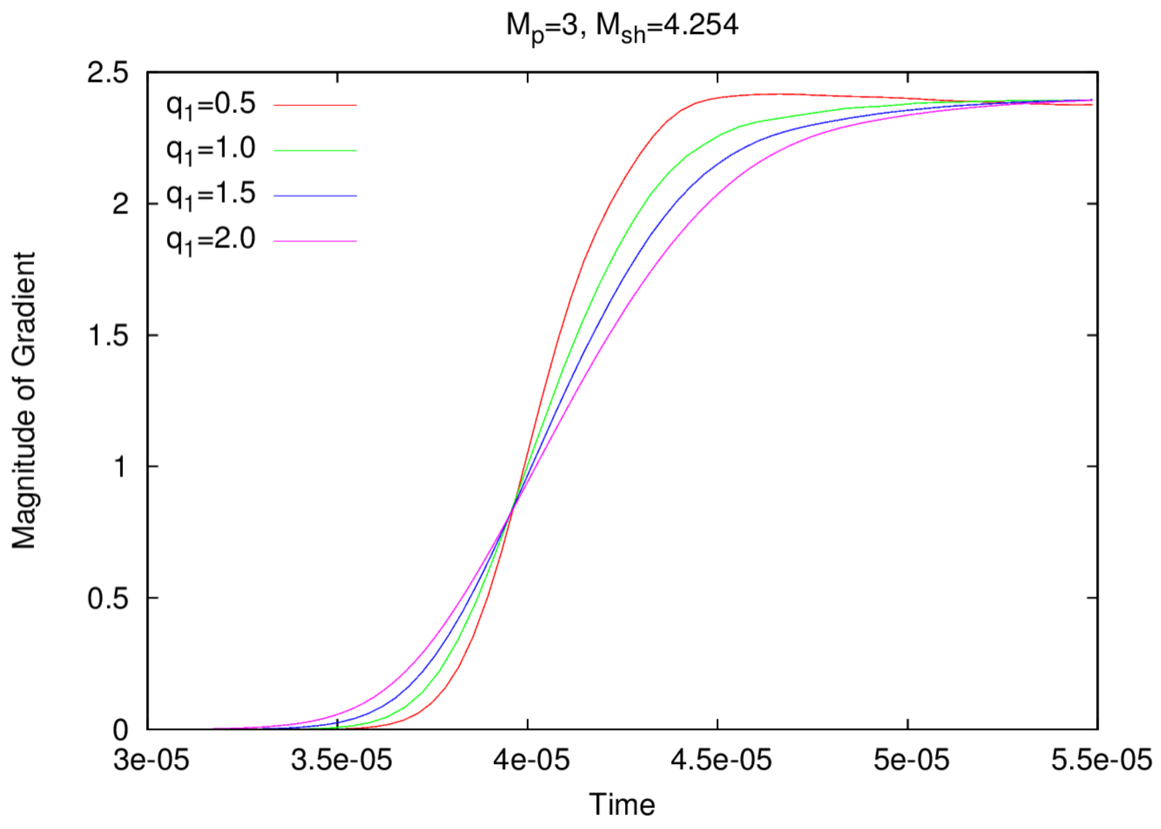


Figure 101: Same as figure 100, but for $M_\mathrm{p} = 3$ and varying the value of q_1. For all runs $q_2 = 1$.

The short nature of this workshop did not allow enough time to fully understand the nature of this potential C function. It did allow time for a preliminary exploration of the parameter space of C from the weak shock limit to the strong shock limit for various combinations of q_1 and q_2, shown in Figure 102. For our 2D piston problem, the time history of $|\nabla x_D|$ is tracked at an initially unshocked node for each run. In post-processing, centered differences are performed on values of $|\nabla x_D|$ from consecutive time steps, which defines $\frac{d}{dt}|\nabla x_D|$ at mid-timestep. A cubic interpolant is used on data near the maximum of $\frac{d}{dt}|\nabla x_D|$ to estimate t_infl. Linear interpolation is then used to find $|\nabla x_D|_{t=t_\mathrm{infl}}$, and thus the value of C.

In the weak shock limit, not much can be said other than the behavior is erratic. It is possible that the above steps taken to estimate the value of the inflection point are not providing a valid answer. More investigation is needed here. In the strong shock limit, there is consistent behavior among all the runs where $q_2 \geq q_1$, all approaching a value ~ 2. Runs done with only q_1 or q_2 had a large amount of non-physical oscillatory behavior of the material parameters in the post-shock region, hence the low values of C since the shock transition ends up overshooting the correct value of $|\nabla x_D|$ initially.

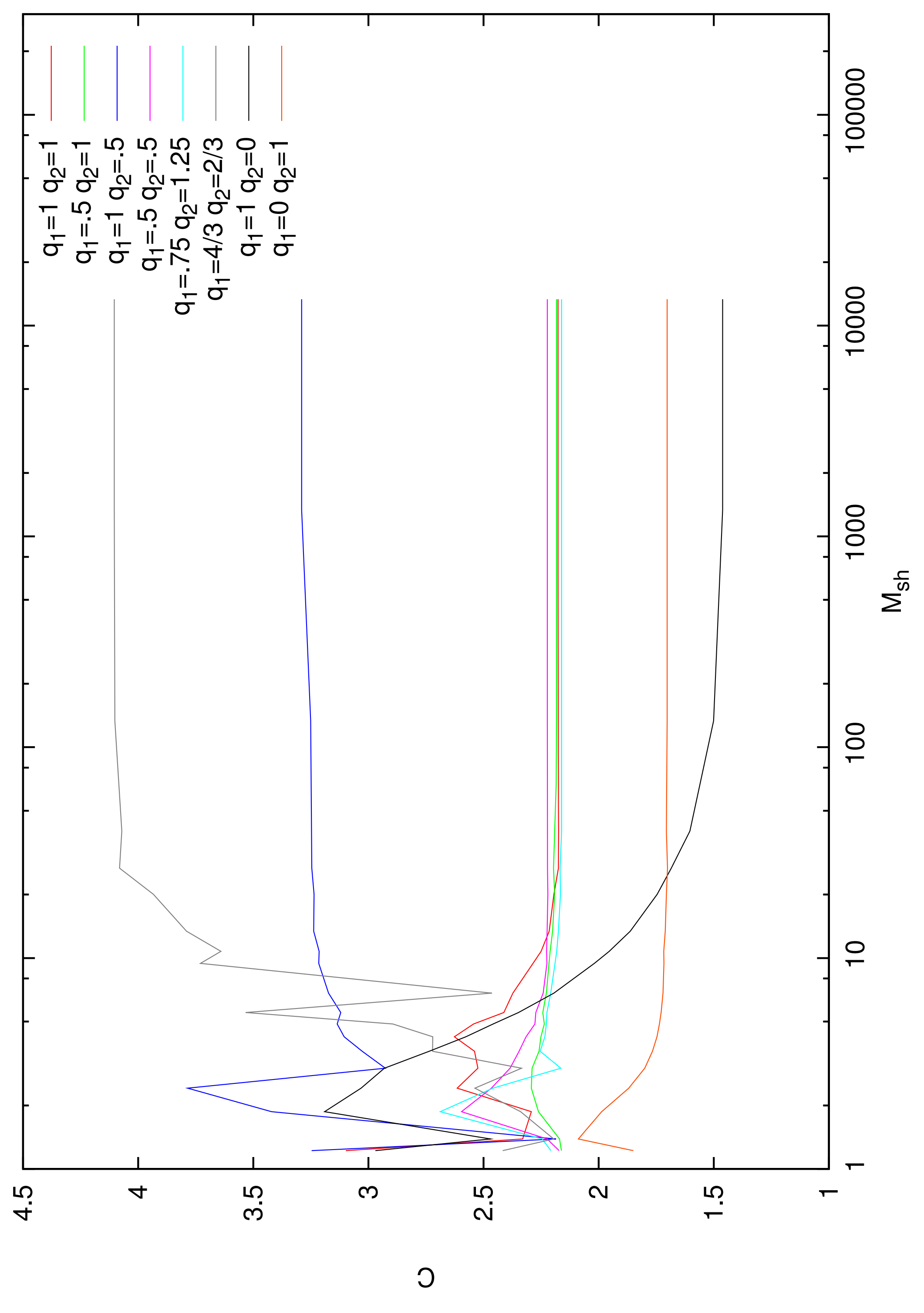


Figure 102: Exploration of parameter space of *C*

Conclusion

The purpose of this work was to devise and implement a method to identify the shock front in two dimensions using the `FLAG` code architecture. The resulting model accomplishes this goal by first calculating the least squares planar gradient of displacement for each node of the mesh. Next, a series of filtering criteria are used to identify valid shock estimates. These estimates are subsequently converted into shock point data structures that can be rendered and visualized in *EnSight*. Finally, a least squares linear fit of the shock point coordinates collected in a zone's stencil is performed in order to identify the shock front interface which is then visualized in *EnSight*.

This entire procedure, from shock estimation to shock reconstruction, has been implemented in `FLAG` and committed to the code repository. The broadcast to initiate the shock profiling routine currently resides in the `lhydro` subroutine, but it can also be called as a user event. The implementation works in parallel and for fully unstructured meshes. The constructed shock fronts can be calculated not only for the purposes of visualization, but for returning shock location data to other physics packages.

Applying a Lagrangian Algorithm to Inertially Confined Fusion Applications

By Peter L. de Vietien, Nathaniel Morgan (Mentor)

Abstract

In this work, a staggered grid Godunov-like Lagrangian hydrodynamic method was used to simulate the compression of an ICF pellet from the University of Rochester's OMEGA facility. The method was tested for its accuracy, its numerical robustness, and its ability to preserve key physical features such as boundary surface instabilities. It was shown that this hydrodynamic method is accurate and more robust than previous methods, and that small surface perturbations were capable of growing due to hydrodynamic instabilities. Unfortunately, the numerical approach used did not perform well on the ICF capsule due to numerical issues arising near the boundaries. The details of the issues are discussed near the end of this work.

Introduction

Experiments were performed at University of Rochester's OMEGA facility on 13 different fuel capsules in order to study the effect of the presence of inert gasses on the ICF neutron yield. The pellet collapse was also simulated, and both the experimental and calculational results were well vetted, but a discrepency between the two persisted. Thus, validation work was sought from LANL, which is the origin of this work [15].

In this pursuit, we attempted to use Lagrangian hydrodynamics, with the Multidirectional Approximate Reimann Solution (MARS) [46], coupled to 3-T diffusion transport equations to simulate the pellet collapse. Along with the hydrodynamics, we used a grey diffusion solver to simulate the radiation transport. The SESAME EOS library was used, and both 1T and 3T physics were studied. The mechanism for energy transfer between the laser and the capsule were quite simple in our simulation, as we skipped the physics of the interaction and directly provided the electrons in the outermost layer with the absorbed energy from the laser.

The OMEGA laser facility combines 60 beams into a 1-ns square pulse that has a total energy of 23 kJ, which amounts to 23 TW [16]. Of the total incident laser energy (23 kJ), a significant portion of the laser energy (40 %) was not absorbed by the capsule. The OMEGA facility is capable of measuring backscattered light, and light that missed the pellet. From these, a profile of the time dependent energy deposition was made. This input was used in the 'experimental' section of this work.

In any computer simulation, whether it be the diffusion of food coloring through water or tracking the density of plasma for nuclear fusion, there are two main parts to checking the performance of the simulation. One part is checking how well the mathematics and computer algorithms that are used in the simulation function; this is called verification. The second part checks how well the physics used in the simulation represent the phenomenon; this is called validation. Verification and Validation (V&V) are the primary tools by which one assesses the accuracy of a simulation, and through V&V we build confidence in computational results.

This paper is organized into three parts. The theory section describes the governing equations for 'pure' hydrodynamics and radiation hydrodynamics. In the Hydrodynamic ICF Simulation section, the accuracy, stability, and the ability to preserve boundary instabilities during the hydrodynamic simulation were tested. Lastly, in the Radiation-Hydrodynamics ICF Experiment section, we implement the three temperature, or 3-T radiation-hydrodynamic equations using a grey diffusion solver and use these to simulate the pellet collapse. Parameters such as material density, opacity, and incoming laser energy from Rochester's OMEGA ICF experiments were included in the rad-hydro simulation. We encountered numerical issues in the rad-hydro simulations, and these issues are explored in this section as well.

Theory

Hydrodynamics

The Lagrangian hydrodynamic equations for conservation of mass, momentum and energy are:

$$\frac{d}{dt}\iint_{\Omega(t)} \rho r dz dr = 0 \tag{119}$$

$$\frac{d}{dt}\iint_{\Omega(t)} \rho \mathbf{u} r dz dr = -\int_{\Sigma(t)} P\mathbf{n} ds \tag{120}$$

$$\frac{d}{dt}\iint_{\Omega(t)} e r dz dr = \iint_{\Omega(t)} -P\nabla\cdot\mathbf{u} r dz dr \tag{121}$$

Here the pressure is calculated by an equation of state, for example a gamma law gas equation of state, $P = (\gamma - 1)\rho e$. The above equations differ from the radiation-hydrodynamic equations which are discussed in the next subsection.

Radiation Hydrodynamics

The rad-hydro equations (122-126) differ from equations (119-121) by the diffusion term $\nabla\cdot(k_x\nabla T_x)$, the source terms (ρs_x), both of which are part of the energy conservation equations, and the pressure terms P_x in the momentum conservation equation, all with x = e for electrons, i for ions, and r for radiation. Because of the separate temperatures for the electrons, ions, and radiation, these equations are called the Three Temperature, or 3-T, equations. The electron, ion, and radiation terms are held together through coupling coefficients ω_{ei} and ω_{er} in equations (127-129). If we choose a low value for these coupling coefficients, then the equations are relatively free to evolve on their own. If we choose a high value, then any small difference in the temperature between the electrons, ions or radiation will result in a large source term, which adds energy to whatever energy term is lowest. Doing this pushes the system towards equilibrium. These temperature coupling equations (127-129) define the source terms s_x in equations (124-126). The simulation assumes that all of the incident laser energy absorbed by the pellet is captured by the electrons, which is why the laser source term s_l is in equation 127 [**?**].

$$\frac{d}{dt}\iint_{\Omega(t)} \rho r dz dr = 0 \tag{122}$$

$$\frac{d}{dt}\iint_{\Omega(t)} \rho \mathbf{u} r dz dr = -\int_{\Sigma(t)} (P_e + P_i + P_r)\mathbf{n} ds \tag{123}$$

$$\frac{d}{dt}\iint_{\Omega(t)} e_e r dz dr = \iint_{\Omega(t)} -(P_e \nabla\cdot\mathbf{u} + \nabla\cdot(k_e \nabla T_e) + \rho s_e) r dz dr \tag{124}$$

$$\frac{d}{dt}\iint_{\Omega(t)} e_i r dz dr = \iint_{\Omega(t)} -(P_i \nabla\cdot\mathbf{u} + \nabla\cdot(k_i \nabla T_i) + \rho s_i) r dz dr \tag{125}$$

$$\frac{d}{dt}\iint_{\Omega(t)} e_r r dz dr = \iint_{\Omega(t)} -(P_r \nabla\cdot\mathbf{u} + \nabla\cdot(k_r \nabla T_r) + \rho s_r) r dz dr \tag{126}$$

And the 3-T Coupling equations are

$$s_e = \omega_{ei}(T_i - T_e) + \omega_{er}(T_r - T_e) + s_l \tag{127}$$

$$s_i = \omega_{ei}(T_e - T_i) \tag{128}$$

$$s_r = \omega_{er}(T_e - T_r) \tag{129}$$

Discrete Approximation

We used a Lagrangian staggered grid hydrodynamics (SGH) Algorithm with a Multidirectional Approximate Reimann Solution [46] to simulate the hydrodynamics. A memetic approach is used to discretize the diffusion terms.

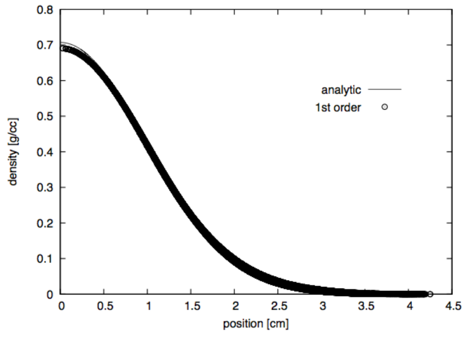
Hydrodynamic ICF Simulation

Before any meaningful simulations can be conducted, the method has to be tested. We want to use a full set of rad-hydro equations to model the pellet collapse, but we start with checking that the hydrodynamic portion works on its own. The hydrodynamic properties that we tested are: The accuracy of the solution; the stability of the mesh; and the ability of the simulation to preserve, and allow the growth of, surface instabilities. After these aspects were tested, radiation transport was added to the hydrodnamic equations. This last part is discussed in the next section.

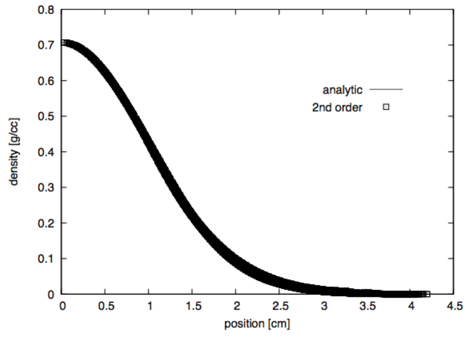
Assessing Simulation Accuracy: Kidder Test Problem

The Kidder test problem simulates the implosion and explosion of a ball of gas. The virtue of the Kidder problem is the availability of an exact solution, from which we can assess the accuracy of the simulation. Key results and details on this test problem are taken from [46] and are quoted below:

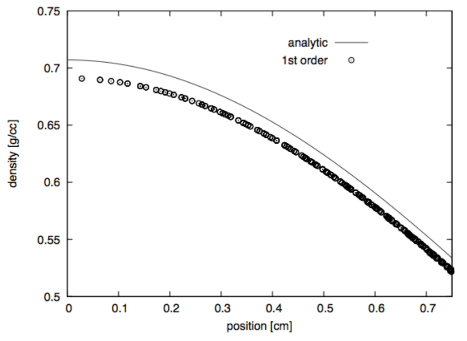
The Kidder RZ problem is a shock-free compression and expansion of a ball of gas. The equation of state is a gamma law gas. The initial conditions of the ball of gas are as follows: an initial inward linear velocity profile, a Gaussian density profile, and a constant internal energy. The velocity profile is $u = \sqrt{r^2+z^2}/2$, *the density profile is* $\rho = \frac{1}{\sqrt{2}} exp(-\frac{r^2+z^2}{2})$, *and the internal energy is* $\frac{3}{8}$. *The gamma of the gas is 5/3. The compressing and expanding ball of gas is modeled on a rectangular* 75×75 *grid where the domain is* 3×3 *cm. The gas is converging on the origin of the mesh, (0,0)... The calculated density and pressure results are compared to the analytic solution at t =2.0s in Fig. 103. As illustrated, the solutions are very symmetric. The results from the 2nd order approach very closely follows the analytic solution. The results from the 1st order approach has noticeable errors near the wall, which are generated by too much dissipation. The 2nd order approach minimizes the dissipation so the calculation is in excellent agreement with the analytic solution.*



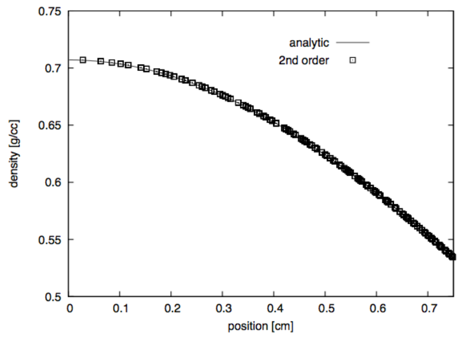
(a) MARS 1st Order



(b) MARS 2nd Order



(c) Zoomed 1st Order



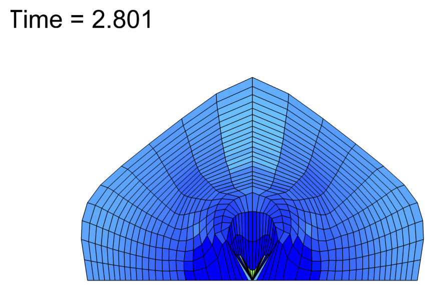
(d) Zoomed 2nd Order

Figure 103: Kidder test problem for 1st and 2nd order MARS. Figures 103c & 103d zoom in on the solution near the origin, where the error is the greatest.

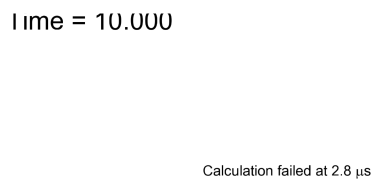
Assessing Mesh Stability: Pressure Ball

The Pressurized Ball problem is used to assess the robustness and symmetry preservation of the algorithm. Key details on this test problem are taken from [46] and are quoted below:

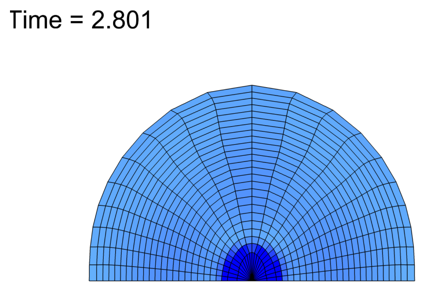
The RZ pressurized ball problem uses a constant pressure boundary condition to compress a sphere of gas. The problem uses a distorted mesh polar mesh to test the symmetry preservation properties of a Lagrangian hydrodynamic scheme. The boundary pressure is 66 2/3 MBar and it is held constant throughout the calculation. The initial state of the gamma-law gas is $\rho = 1g/cc$ *and* $e = 1MBarcc/g$*. The problem uses a gamma of 5/3... The initial outer radius of the gas sphere is 10 cm. The computation mesh has 24 uniform radial cells, and 9 non-uniform angular cells... The algorithm will preserve symmetry on an equal angle polar mesh, but the non-uniform angular discretization causes a loss of symmetry in the solution... The pressurized ball problem is difficult to calculate with SGH methods and calculations can fail prior to 5 s. The results from the [Von Neumann Richtmeyer] VNR and the Riemann-like approaches are provided in Fig.* **??** *for the RZ pressurized ball problem. The VNR approach used the TQS mesh stability model. The calculation with VNR viscosity failed at 2.8 μs. In contrast, the 1st order and 2nd order accurate Riemann-like approaches were able to reach a time of 10 μs... As mentioned before, we believe the Riemann-like approach is dissipating the directional errors that are created in SGH with non-uniform zoning. Increasing the dissipation smoothes the mesh, but it could also dissipate physical vorticity. The RZ pressurized ball problem illustrates the Riemann-like approach is reasonably robust without additional mesh stability models.*



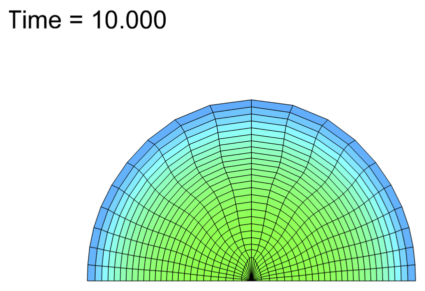
(a) VNR



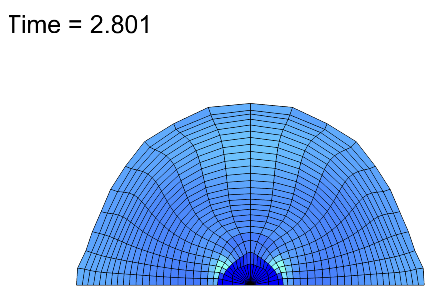
(b) VNR



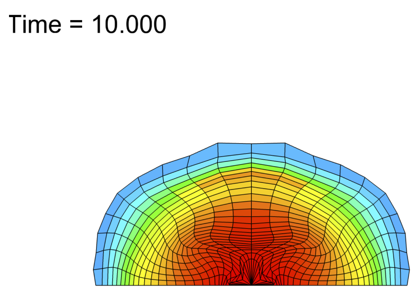
(c) MARS 1st Order



(d) MARS 1st Order



(e) MARS 2nd Order

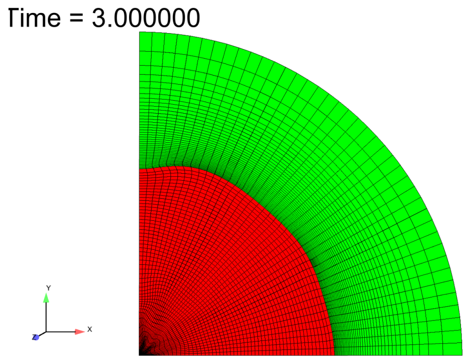


(f) MARS 2nd Order

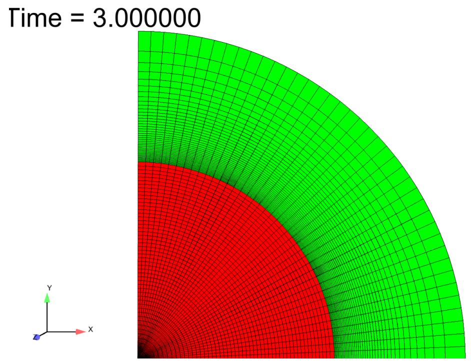
Figure 104: Mesh Stability Analysis for VNR and MARS approach: (a),(b): VNR approach leads to solution crashing and mesh tangling. (c),(d): MARS 1st Order delivers a stable mesh, with (e), (f) MARS 2nd order proving a less rigid mesh.

Assessing Ability to Preserve Physical Features: Surface Instabilities

A key physical feature that needs to be simulated is instability on the fuel surface boundary. As the outer layer of material is rocketed inwards and compressing the pellet, instabilities commonly develop which can limit the amount of compression of the pellet. For our current purposes, we do not need to replicate these instabilities, that can be done more rigorously in future work, but we do need to demonstrate that the hydrodynamic approach is capable of allowing these instabilities to exist and grow. As can be seen in Fig. 105, the 2nd order MARS approach allows these instabilities, whereas the first order approach does not.



(a) MARS 2nd Order



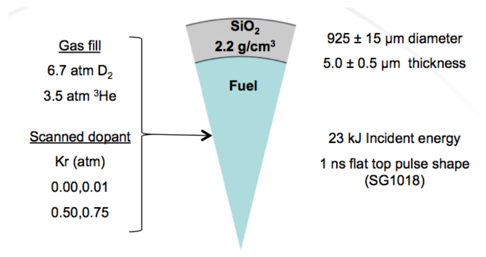
(b) MARS 1st Order

Figure 105: The 2nd order MARS approach shows perturbation growth, and the 1st order approach does not. An initial perturbation amplitude of $10^{-}4$ was implemented, and the final perturbation amplitude was on the order of 10. An allowance of 10^5 growth in the perturbation implies that the hydrodynamic approach is worth looking into further.

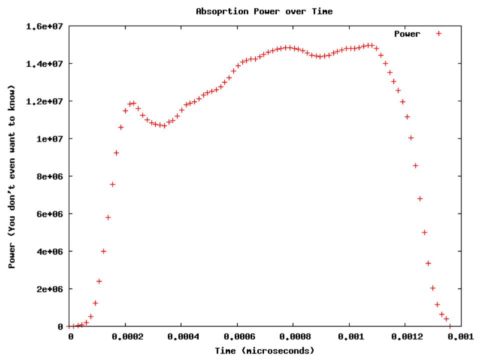
Radiation-Hydrodynamics ICF Experiment

Now that we have verified the accuracy of the solution and the stability of the mesh, and validated that the simulation can run to completion and preserve the physical phenomena of a surface instability, we will implement a radiation transport solver into the simulation, which provides us with a radiation hydrodynamic code.

The capsule that was used in the OMEGA experiments is shown in Fig. 106a below. The capsule has an outer radius of 925 $\pm$ 15 μm and a thickness of 5.0 $\pm$ 0.5 μm, and a 2.2 g/cm^3 average density. The capsule contained a fuel mixture of 6.7 atm of deuterium and 3.3 atm of helium-3. No fusion related properties were taken into account for these simulations, which is okay as this simulations reaction rate is low enough to not qualitatively impact the hydrodynamics. These experimental parameters were not key in this work, since our first focus was to test the applicability of a rad-hydro MARS method to the pellet collapse.



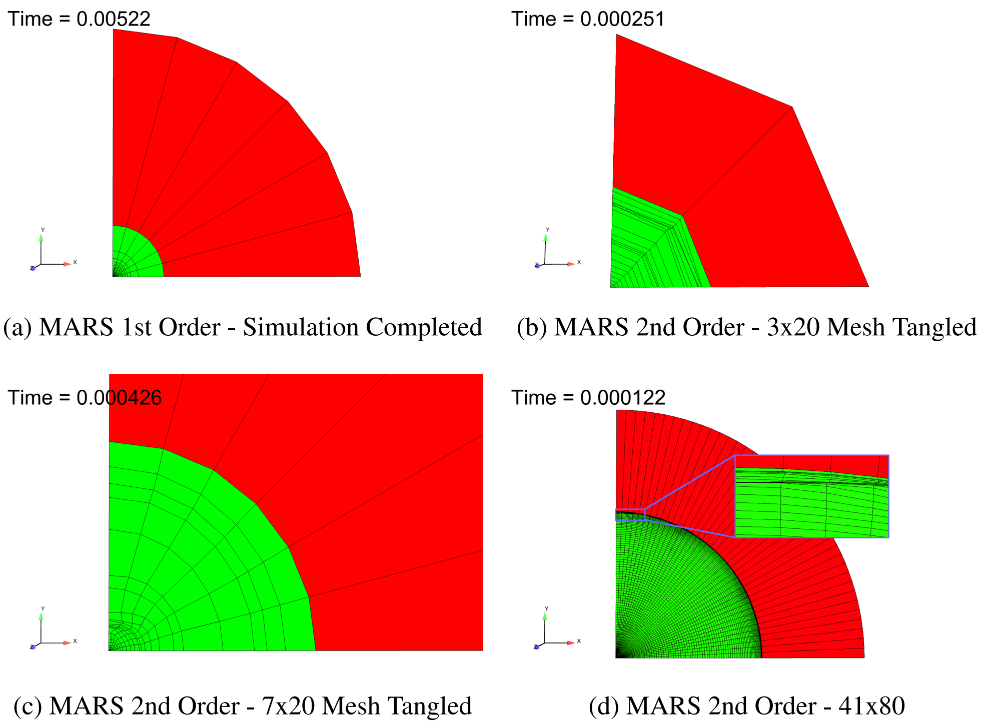
(a) Capsule geometry and composition



(b) Absorbed power profile on outer layer of capsule.

Figure 106: Capsule geometry, composition and temporal power absorption

When using MARS with the rad-hydro equations, negative mesh volumes would result. This only occurred for the Rad-Hydro equations, not the pure hydro equations. The first order simulation 107a could run for some mesh topologies, but not all. The second order MARS approach failed for almost all mesh topologies, and for all values of q1n, which indicated the extent of energy dissipation used in the simulation. We tried different mesh resolutions with the second order approach (107b,107c,107d) which led to the simulation failing at different times. Similar results were obtained for strong and weak coupling, or 1-T and 3-T equations.



(a) MARS 1st Order - Simulation Completed (b) MARS 2nd Order - 3x20 Mesh Tangled

(c) MARS 2nd Order - 7x20 Mesh Tangled (d) MARS 2nd Order - 41x80

Figure 107: Rad-Hydro solutions leading to tangled mesh's for MARS 2nd order, running completely for 1st order.

In order to investigate why the simulation was crashing, we ran simulations of just the radiation diffusion solver. Since there was no hydrodynamic component of these simulations, the mesh was static. We ran the radiation diffusion problem for both weak and strong coupling (1 and 10^4 for the coupling coefficients ω_{ei} and ω_{er}), and the simulation ran completely for both cases. An interesting feature is the small temperature gradients present in results, which occurred even at low coupling. From this, we established that the radiation transport and the second order MARS pure hydrodynamics works independently, and the simulation crashes only when both are ran together.

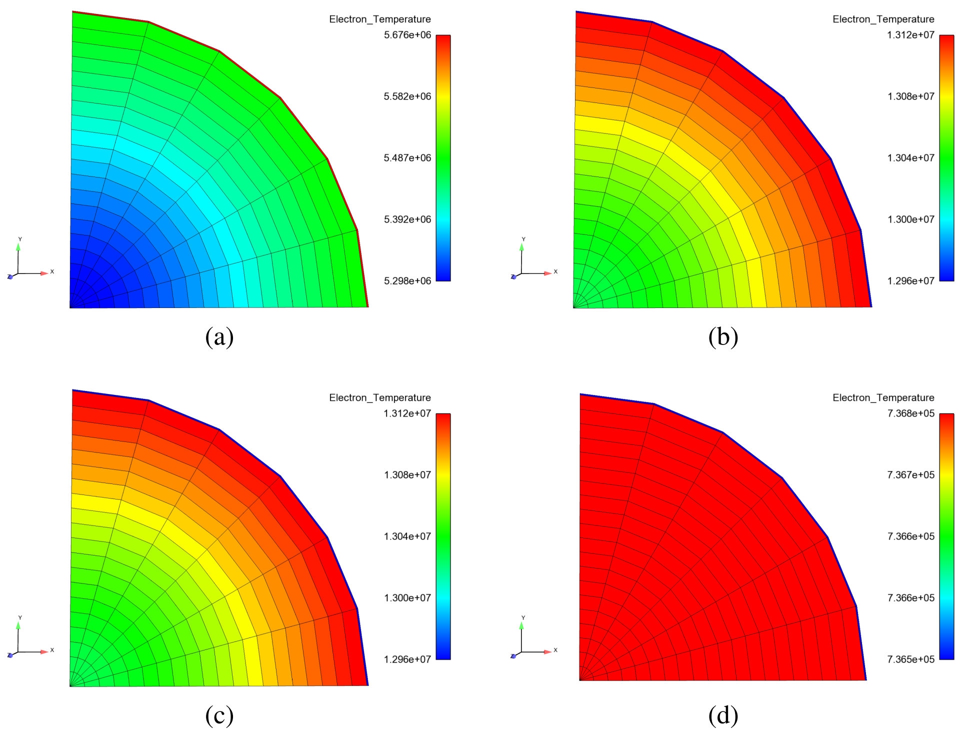


Figure 108: Rad-Hydro solutions leading to tangled mesh's for MARS 2nd order, running completely for 1st order.

Lastly, a study was performed to asses the impact of the coordinate system on numerical errors. To test this, we ran the simulation in cartesian coordinates instead of the original cylindrical RZ coordinates. Using Cartesian coordinates, both first and second order MARSs method rad-hydro simulations ran completely. From this, we know that the simulation issue occurs under the conditions of using the second order MARS full rad-hydro equations in RZ coordinates. The relatively successful cartesian simulation is shown in 109.

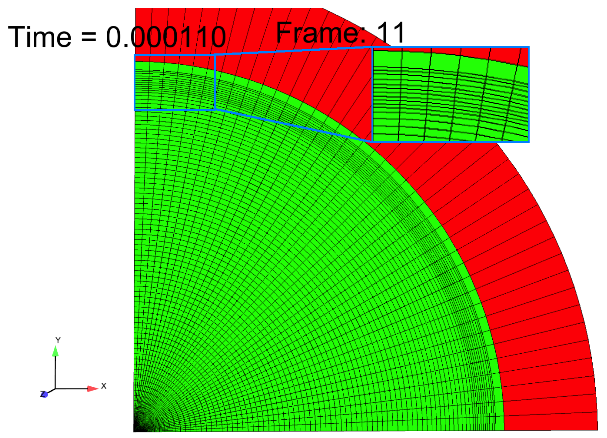


Figure 109: Cartesian coordinate calculation reaches completion with both MARS approaches.

Conclusion

The Lagrangian SGH Godunov-like method was studied here on problems of relevance to ICF. The first three test problems were chosen to test the applicability of the hydrodynamic algorithm to the ICF capsule collapse. The test problems are: The Kidder problem, the Pressure Ball problem, and the Surface Instability problem. The SGH Godunov-like method can accurately solve the Kidder problem, with the second order approximation producing a more accurate result than the first order solution. The method was also more numerically stable than previous methods for the pressure ball problem. Lastly, the method preserves and allows the growth of perturbations. This is important as its numerical stability from the pressure ball problem could imply that it would completely suppress these physical features, but it does not.

Since the hydrodynamics tests above produced positive results, we proceeded to test 3-T and 1-T radiation-hydrodynamic equations using the MARS method. The results suggest that there are numerical issues near the r = 0 boundary with the MARS method in cylindrical RZ coordinates. The numerical issue is not present with 'pure' hydro, nor with just diffusive radiation transport, and does not depend on the coupling for the 3-T equations. Lastly, the simulation runs completely when using Cartesian coordinates. Future work should investigate the issues causing the MARS method to crash under these conditions.

3D Lagrangian Hydrodynamics

Student Author:
ROBERT A. ZIMMERMAN

Mentor:
Dr. NATHANIEL R. MORGAN

Abstract

Much research in Lagrangian staggered-grid hydrodynamics (SGH) has focused on explicit viscosity models for adding dissipation to a calculation that has shocks. The explicit viscosity is commonly called artificial viscosity. In a previous paper by Morgan et al. 2012 an approach that adds dissipation to the calculation via solving a Riemann-like problem using a Multidimensional Approximate Riemann Solution (MARS) was presented. The current paper extends that work from 2 dimensional problems to 3 dimensions. The approach is valid for many complex multidimensional flows with strong shocks. Numerical details and test problems are presented. The test problems include a comparison to known analytical solutions and are verified using a convergence study with good result.

Introduction

The project was completed by the student group under mentor Nathaniel R. Morgan Ph.D. in the 2014 Computational Physics Student Summer Workshop under the direction of Scott R. Runnels Ph.D. In XCP-4 (Computational Physics Division, group 4). The group's project consisted of testing a new Multidimensional Approximate Riemann Solution (MARS) algorithm inside of the FLAG (Free LAGrangian) program commonly utilized and maintained by Los Alamos National Laboratory (LANL). The particular method application is a 3D Lagrangian staggered grid Godunov-like approach for hydrodynamics. The objective of the summer was to find the breaking points of the algorithm based on test cases with known analytic solutions that the numeric results can be compared to, as well as verify the 3D convergence based on improving the lagrangian grid resolution. Numerical results are calculated with first and second order approximations. Though first order methods were succesful, second order methods were also tested to create improved accuracy of the calculated solutions.

The Lagrangian numerical approach is commonly used to calculate shock wave problems. The Lagrangian approach solves the conservation equations on a mesh that moves with the flow. It is common to solve the discrete governing equations on two different control volumes, such a method is called staggered grid hydrodynamics (SGH). The SGH method solves the conservation of momentum equation on a control volume around the node, where the nodal control volumes are referred to as the dual grid. Likewise, the change in the internal energy equation is solved on a control volume that coincides with the cell boundary. The velocity is located at the vertex of a cell, termed a node, and the thermodynamic variables are located at the cell center. A MARS is found at the cell center and accounts for discontinuities in the flow (e.g. shocks). MARS will add dissipation to the solution which provides numerical stability and increases the entropy.

Background Theory

Governing Equations

The Governing equations describing the selected test problems are

$$\text{Mass} \quad \frac{\mathrm{d}M}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}\int\limits_V \rho \mathrm{d}V = 0 \tag{130}$$

$$\text{Volume} \quad \frac{\mathrm{d}V}{\mathrm{d}t} = \int\limits_V u \cdot \mathrm{d}A \tag{131}$$

$$\text{Momentum} \quad \frac{\mathrm{d}(Mu)}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}\int\limits_V \rho u \mathrm{d}V = \oint\limits_{\partial V} (\mathrm{d}N \cdot \sigma) \tag{132}$$

$$\text{Energy} \quad \frac{\mathrm{d}(MJ)}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}\int\limits_V \rho\left(e + \frac{u^2}{2}\right)\mathrm{d}V = \oint\limits_{\partial V} (\mathrm{d}N \cdot \sigma \cdot u) \tag{133}$$

$$\text{Velocity} \quad \frac{\mathrm{d}x}{\mathrm{d}t} = u \tag{134}$$

The discrete form of the governing equations are expressed in terms of summations and discrete derivatives.

$$\text{Mass} \quad \frac{\Delta M}{\Delta t} = 0 \tag{135}$$

$$\text{Volume} \quad \frac{\Delta V_z}{\Delta t} = \sum_{ECV} (u_p \cdot nA)^{n+\frac{1}{2}} \tag{136}$$

$$\text{Momentum} \quad M_p \frac{\Delta u_p}{\Delta t} = \sum_{MCV} \left(\sigma_z^{total} \cdot nA\right)^{n+\frac{1}{2}} \tag{137}$$

$$\text{Energy} \quad M_z \frac{\Delta e_z}{\Delta t} = \sum_{ECV} \left(\sigma_z^{total} \cdot nAu_p\right)^{n+\frac{1}{2}} \tag{138}$$

$$\text{Velocity} \quad \frac{\Delta x_p}{\Delta t} = u_p^{n+\frac{1}{2}} \tag{139}$$

$$\text{Other} \quad \sigma_z^{total} = \sigma_z - q_z^* \tag{140}$$

Lagrangian and Eulerian Mesh Descriptions

In computational hydrodynamics there are two basic methods of describing fluid motion. They are Lagrangian and Eulerian descriptions. The Eulerian description is from a fixed frame of reference and material is observed as it passes through space. Here the observer is the fixed reference point watching the fluid pass. In computational hydrodynamics this method is achieved by creating a fixed mesh that does not move, then the transport equations are solved to determine the amount of material moving between the fixed mesh cells. The Lagrangian description is with a reference point that is fixed to the material that moves through space and deforms with the continuum. The idea of Lagrangian versus Eulerian descriptions as they are applied to meshes and the MARS method is illustrated below in figures 111 and 110 respectively.

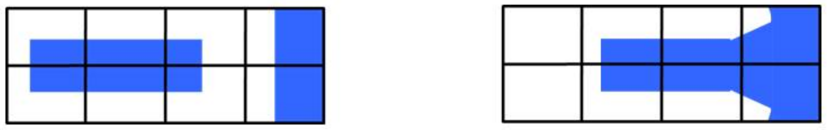


Figure 110: Eulerian Mesh Description

Figure 110 shows an Eulerian mesh (stationary) with the material advecting and diffusing through the mesh. Note that the Eulerian mesh doesn't deform or move.

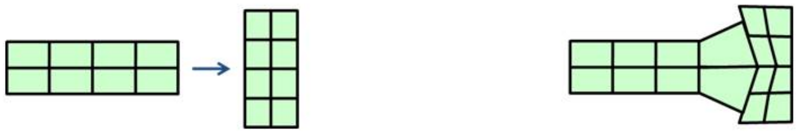


Figure 111: Lagrangian Mesh Description

Figure 111 shows a lagrangian mesh deforming with the material. It can be seen that the amount of material inside of a cell does not change.

Staggered Grid Hydrodynamics

The staggered grid hydrodynamics method is a spatially staggered method that solves the governing equations on different (or staggered) control volumes. The particular SGH method used here is illustrated in figure 112 (2D). The density, pressure, and internal energy are stored at the cell center and the velocity is stored at the nodes. The MARS is at the cell center and the details are provided in [Morgan et. al 2012], relevant details are also quoted in section

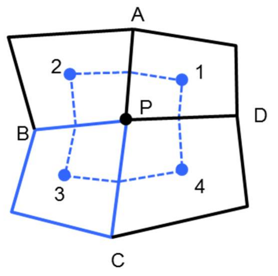


Figure 112: SGH. Solid blue is the cell boundary, dashed blue is the nodal control volume, solid black is the mesh

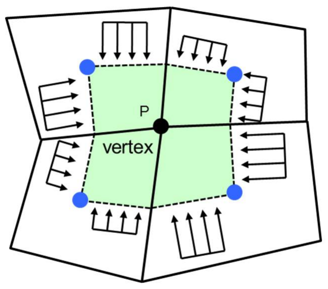


Figure 113: SGH

Figure 113 shows how the acceleration of the node (132) is calculated on the nodal control volume using the forces from the neighboring cells.

Methodology

The details for second order approximations, limiters, the Riemann-like Solution, time integration, and the boundary conditions applied to this problem are discussed in the following sections.

Second Order Approximation

Second order accuracy is achieved by reconstructing the velocity field with linear Taylor series expansions. The motivation for second order accuracy and the details on the Taylor series are quoted from [Morgan and Burton et. al 2012].

The goal of the second-order extension is to regulate the amount of dissipation added to the calculation. Specifically, we seek to only include dissipation around shocks. Regulating the dissipation is not a new conceptin SGH (staggered Grid Hydrodynamics). For example, Christensen, Benson, Caramana, and Loubre used limiters to regulate the explicit viscosity terms.

The dissipation is regulated by multiplying the explicit viscosity terms by $1-\phi$, where ϕ is a limiting coefficient and $\phi \in [0:1]$. A value of 1 eliminates the explicit viscosity terms and a value of 0 uses the explicit dissipation terms in the calculation. Limiting the dissipation in SGH can be problematic. A study by Loubre illustrates the viscosity limiters can cause extremely poor results. The approach here differs from the SGH limiting approaches used by Christensen, Benson, Caramana, and Loubre. The approach employed in this work limits the amount of dissipation by reducing the jump in the velocity. The Riemann-like problem at the cell center requires a velocity. The first-order accurate approach uses the velocity at the node; whereas, the second-order accurate approach projects the velocity from the node to the cell center via a linear Taylor-Series expansion. The projection is therefore given by

$$u_c = u_p + \phi \cdot r_{zp} \cdot \nabla u_p \tag{141}$$

The velocity projection to the cell center in the principal strain direction, as calculated at the node, is given by

$$u'_c = u'_p + \phi' \cdot r_{zp} \cdot \nabla u'_p \tag{142}$$

where the prime denotes the principal direction and ϕ' is a diagonal tensor with limiting coefficients. The limiting coefficients are calculated using the BarthJespersen method when the limiting tensor is rotated in to the principal strain direction. The BarthJespersen limiter is used to compare a projected velocity, which is rotated into the principal direction, with the maximum and minimum velocities at the neighboring nodes. The maximum and minimum velocities at the neighboring nodes are also rotated into the same principal direction as the projected velocity.

Limiters

The Barth−Jespersen limiter is defined by

$$\phi' = \begin{cases} f\left(BJ_{fac}\frac{(u'_{max}-u'_p)}{u'_c-u'_p}\right) & \text{if } u'_c > u'_p \\ f\left(BJ_{fac}\frac{(u'_{min}-u'_p)}{u'_c-u'_p}\right) & \text{if } u'_c < u'_p \\ 1 & \text{if } u'_c = u'_p \end{cases} \tag{143}$$

where

$$f(r) = min\left(BJ_{max}, r, 1\right) \tag{144}$$

The limiting process is directly quoted from [Morgan and Burton et. al 2012].

The prime denotes a velocity in the principal direction and ϕ' is a limiting coefficient in the diagonal tensor ϕ'. The max/min subscript is the maximum/minimum neighboring nodal velocities in the same principal direction as calculated at the node p. A limiting coefficient is found for each velocity component in the principal direction. The limiting tensor is rotated out of the principal strain direction of the node

Riemann-like Solution

The Riemann-like solution can be characterized by a five step process.

Step (1) project the velocity to the cell center using a first or second order assumption.

$$\text{First Order Approximation} \quad \hat{u}_c = u_p \tag{145}$$

$$\text{Second Order Approximation} \quad \hat{u}_c = u_p + r_{pz} \cdot \bar{\nabla} u_p \tag{146}$$

where $\bar{\nabla} u_p$ is the limited gradient

step (2) calculate a unit vector that approximates the direction of the shock.

$$a_c = \frac{u_p - u_z}{||u_p - u_z||} \tag{147}$$

$$u_z = AVG(u_p) \tag{148}$$

$$\tag{149}$$

step (3) Calculate the normal vectors of all the nodal control volume surfaces inside the cell. These will be used to calculate Riemann forces.

$$\sum_{faces} F = \sum_{faces} \sigma_i^* \cdot N_i = 0 \tag{150}$$

$$F_c = \sigma_z^* \cdot n A_c \tag{151}$$

$$\sigma_c^* = \sigma_z + q_z^* \tag{152}$$

$$q_c \cdot n = (\sigma_c^* - \sigma_z) \cdot n \tag{153}$$

$$q_c \cdot n = \mu \delta u_z^* \tag{154}$$

The dissipation relationship says the jump in stress is proportional to the jump in velocity

$$\mu = \rho U \quad \text{Shock Impedance} \tag{155}$$

$$U = b_0 + b_1 \delta u_c^* \quad \text{Shock Velocity} \tag{156}$$

$$q_c \cdot n = (\rho a + \rho b_1 |\delta u_c|)\, \delta u_c^* \tag{157}$$

$$|\delta u_c| = |u_z - u_c| \tag{158}$$

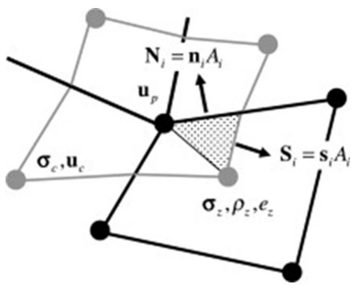

$\mathbf{N}_i = \mathbf{n}_i A_i$
$\mathbf{u}_p$
$\mathbf{S}_i = \mathbf{s}_i A_i$
$\sigma_c, \mathbf{u}_c$
σ_z, ρ_z, e_z

Figure 114: SGH

Figure 114 shows the normals being calculated

step (4) Calculate Riemann velocity

$$u_z^* = \frac{\sum\limits_{i \in z} \left[A_i \big(\mu_{c(i)} |a_{c(i)} \cdot s_i| u_{c(i)} - \sigma_z \cdot s_i \big) \right]}{\sum\limits_{i \in z} \left[A_i \mu_{c(i)} |a_{c(i)} \cdot s_i| \right]} \tag{159}$$

The Riemann velocity equation (159) is found by applying the dissipation relations to each face in the cell and enforcing conservation of momentum (150) to find the Riemann velocity. A 2D example is provided of the 8 dissipation relations that would exist for a quadrilateral cell. The system of equations is closed by using (150) because there are 9 unknowns and now 9 equations for the 2D example.

$$\mu \left(u_z^* - u_1 \right) |a_1 \cdot s_{12}| = \left(\sigma_1^* \cdot s_{12} - \sigma_z \cdot s_{12} \right) \tag{160}$$
$$\mu \left(u_z^* - u_1 \right) |a_1 \cdot s_{14}| = \left(\sigma_1^* \cdot s_{14} - \sigma_z \cdot s_{14} \right) \tag{161}$$
$$\mu \left(u_z^* - u_2 \right) |a_2 \cdot s_{21}| = \left(\sigma_2^* \cdot s_{21} - \sigma_z \cdot s_{21} \right) \tag{162}$$
$$\mu \left(u_z^* - u_2 \right) |a_2 \cdot s_{23}| = \left(\sigma_2^* \cdot s_{23} - \sigma_z \cdot s_{23} \right) \tag{163}$$
$$\mu \left(u_z^* - u_3 \right) |a_3 \cdot s_{32}| = \left(\sigma_3^* \cdot s_{32} - \sigma_z \cdot s_{32} \right) \tag{164}$$
$$\mu \left(u_z^* - u_3 \right) |a_3 \cdot s_{34}| = \left(\sigma_3^* \cdot s_{34} - \sigma_z \cdot s_{34} \right) \tag{165}$$
$$\mu \left(u_z^* - u_4 \right) |a_4 \cdot s_{43}| = \left(\sigma_4^* \cdot s_{43} - \sigma_z \cdot s_{43} \right) \tag{166}$$
$$\mu \left(u_z^* - u_4 \right) |a_4 \cdot s_{41}| = \left(\sigma_4^* \cdot s_{41} - \sigma_z \cdot s_{41} \right) \tag{167}$$

step (5) calculate viscous force

$$q_i^* = \mu_c \left(u_z^* - u_c \right) |a_c \cdot S_i| \tag{168}$$

Time Integration

The time integration uses the same approach discussed [Morgan and Burton et. al 2012] and key details from the paper are quoted.

The momentum and compatible internal energy equation are integrated in time via a two step process. The first step is to obtain a solution at $n + \frac{1}{2}$, and the second step is to calculate the solutionat $n + 1$. The details of these two steps are as follows. The time integration begins with projecting the solution forward in time by $\frac{\Delta t}{2}$. The projection step only uses information at time n. For example, the mesh position at $n + \frac{1}{2}$ is $x^{n+\frac{1}{2}} = x^n + \frac{\Delta t}{2} u^n$. Next, the Riemann forces are calculated using the solution estimate at $n + \frac{1}{2}$. These forces are used in both the momentum and compatible internal energy equations. The second integration step is identical to the central difference integration approach, where the right hand side of Eqs.(1),(2),and(3)are at $n + \frac{1}{2}$. The momentum equation is integrated in time from n to $n + 1$ using the Riemann forces at $n + \frac{1}{2}$. The compatible internal energy equation is integrated in time from n to $n + 1$ using the *temporal average velocity, $u_p^{n+\frac{1}{2}} = \frac{1}{2}(u_p^{n+1} + u_p^n)$, and the Riemann forces at $n+\frac{1}{2}$. The mesh position is integrated in time from n to $n+1$ using the temporal average velocity. All values are now known at $n+1$. The integration process is repeated until the calculation is complete.*

Boundary Conditions

The boundary conditions use the same approach discussed [Morgan and Burton et. al 2012] and key details from the paper are quoted.

The solution at boundaries must take into account the appropriate physical constraints. The two boundary conditions considered here are a reflected boundary and a stress boundary. The approaches presented here will preserve symmetry on equal angle polar meshes with a radial flow, which is a design goal. The discussion will first focus on the reflected boundary condition followed by the stress boundary condition. The reflected boundary condition imposes the appropriate physical constraints for a symmetry plane, a wall, or a piston. The physical constraints at a reflected boundary are the gradient in both the stress and velocity are equal to zero in the surface normal direction. The control volume for the momentum equation and the velocity gradients at a reflected boundary is identical to an internal node. To enforce these constraints, the mesh is reflected in the surface normal direction, and the boundary node is treated like an internal node. The velocity gradient and the stress gradient will be equal to zero in the surface normal direction. The stress boundary condition imposes the appropriate physical constraints for a free surface (e.g. zero pressure) or a boundary stress (e.g. pressure). This condition is enforced by using a control volume for the momentum equation and the velocity gradient(Eqs.(17),(18))that includes the boundary surface. The boundary stress, σ_b, is applied to the control volume along the boundary. These boundary forces, $S_{i(b)} \cdot b$, are included in the momentum equation just like other forces. For a free surface, the boundary forces are equal to zero because the pressure is equal to zero. The velocity along the boundary is equal to the nodal velocity. The boundary contributions for the velocity gradient are $S_{i(b)} u_p$. The velocity gradient(Eqs.(17),(18)) includes these boundary contributions.

Corner Pressures

The Multidirectional Riemann problem can include corner pressures. The corner pressures are calculated by a thermodynamic expansion along an isentrope from the cell center properties. The corner pressures are calculated by

$$p_c = p_z + \alpha_{hg}\left(p_c^\circ - p_z\right)$$

where

$$p_c^\circ = fcn\left(c_z, \rho_c, \rho_z\right)$$

L1 Convergence

The L1 error norm is also known as the least absolute deviations regression. It is defined as $S = \frac{1}{n}\sum_{i=1}^{n} |y_i - f(x_i)|$ where y_i is the analytical solution and $f(x_i)$ is the numerical solution. The convergence script is also listed in the appendix along with an example Flag input file.

Procedures

Problems and Parameters

Five 3D Cartesian test problems were used in this study: Noh-3D, Sedov-3D, Kidder-3D, Verney-3D, and Taylor Anvil. The 3D mesh resolution used with Noh, Sedov, and Kidder are 20x20x20, 40x40x40, 60x60x60, 80x80x80, and 100x100x100 cells. A single resolution was used with Verney and Taylor Anvil because no convergence studies were performed on these two problems. This test suite was designed to quantify accuracy on a range of problems covering gas dynamics and problems involving strength. Both smooth flows, vortical flows, and radial flows with strong shocks are studied. Each test problem was run exploring a range of parameters including first order, second order, second order with the corner pressure α parameter set to 0.1, 0.5, 1.0 and some of the problems were run in second order with the Barth-Jesperson limiters set to $bj_{max} = 0.5$ and $bj_{fac} = 1$. Each parameter studied was set individually and not in combination with any others. The spreadsheet describing parameter variation is included.

		Mesh Size				
test problem	Order	20x20x20	40x40x40	60x60x60	80x80x80	100x100x100
Sedov-3D	1st Order	y	y	y	y	y
	2nd Order	n	n	n	n	n
	Alpha = 0.1	y	y	y	y	y
	Alpha = 0.5	y	y	y	y	y
	Alpha = 1	y	y	y	y	y
	Bjmax = 0.5	N/A	N/A	N/A	N/A	N/A
	Bjfac = 1	N/A	N/A	N/A	N/A	N/A
Sedov-RZ	1st Order	y	y	y	y	y
	2nd Order	y	y	y	y	y
	Alpha = 0.1	y	y	y	y	y
	Alpha = 0.5	y	y	y	y	y
	Alpha = 1	y	y	y	y	y
	Bjmax = 0.5	y	y	y	y	y
	Bjfac = 1	y	y	y	y	y
Noh-3D	1st Order	y	y	y	y	y
	2nd Order	y	y	y	y	y
	Alpha = 0.1	y	y	y	y	y
	Alpha = 0.5	y	y	y	y	y
	Alpha = 1	y	y	y	y	y
	Bjmax = 0.5	y	y	y	y	y
	Bjfac = 1	y	y	y	y	y
Kidder-3D	1st Order	y	y	y	y	y
	2nd Order	y	y	n	n	n
	Alpha = 0.1	y	y	y	y	n
	Alpha = 0.5	y	y	y	y	y
	Alpha = 1	y	y	y	y	y
	Bjmax = 0.5	N/A	N/A	N/A	N/A	N/A
	Bjfac = 1	N/A	N/A	N/A	N/A	N/A
Kidder-RZ	1st Order	y	y	y	y	y
	2nd Order	y	y	y	y	y
	Alpha = 0.1	N/A	N/A	N/A	N/A	N/A
	Alpha = 0.5	N/A	N/A	N/A	N/A	N/A
	Alpha = 1	N/A	N/A	N/A	N/A	N/A
	Bjmax = 0.5	N/A	N/A	N/A	N/A	N/A
	Bjfac = 1	N/A	N/A	N/A	N/A	N/A

All problems succeeded using the first order approximation, however, not all problems succeeded with second order accuracy without limiters or parameter variation. The second order method allows more accuracy and is thus of the most interest to this paper. The limiters were varied to enable successful completion of the problem runs.

Computer Systems and Programs

The problems were computed on the Moonlight and Mapache supercomputer systems using between 16 to 128 processors for up to 4 hours of computing time. All calculations, runs and data analysis were completed using the unix systems RedHat 5 and 6 on the Sunray computer system. Part of the summer training and learning consisted of learning how to navigate and operate a unix operating system as well as learning the Python , C++, and AJAX programming languages. Images of the problems were then taken at several angles for the mesh grid 40x40x40 for first and second order unless the second order run failed in which case it was taken at a corresponding successful alpha value or BarthJesperson limiter. Three parameters were recorded for the images: pressure, energy, density. Images were also taken at three time steps: t_start, t_mid, t_end. This varied from problem to problem but t_start always correlated to the first time step (t= 0), t_end correlated to the final time step (t= 0.6 for Noh-3D) and t_mid correlated to the median time step (t=0.3 for Noh-3D). See spread sheet below for time steps and image angles.
The program used to visualize the problems and take images was Ensight 10.0. Though this program worked very well an error was encountered using the Sunray computer systems in conjunction with the Ensight program. Images were being produced with phantom green reflections (line patterns resembling the object, although skewed in position and orientation), to overcome the problem it was discovered after consulting with High Performance Computing (HPC) that opening Ensight with the -X option suppressed this effect. A python script was also written for the process of taking images and selecting parameters, as well as choosing colors and legend spacing. The program is listed in the appendix. Scatter plots of mesh grids 40x40x40, 60x60x60, and 100x100x100 are matched with an analytic solution for each considered parameter (density, energy, and pressure) and plotted. These plots were created using a Gnuplot script to visualize how close the numeric solution matches the analytic solution. The Gnuplot script is listed in the appendix. Convergence is then evaluated using a python script that considers each grid refinement using an L1 error norm.

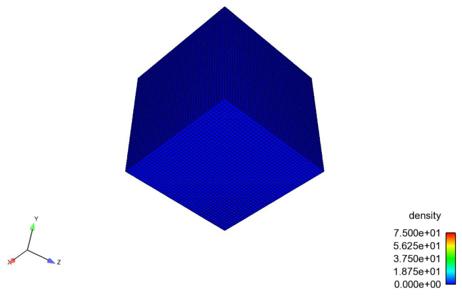
Problem Results

All the test problems except Taylor Anvil were chosen because they have an analytical solution that numerical solutions can be compared to. The Taylor Anvil Problem has experimental data for comparing the numerical solution. They are also selected because they have been known to break many hydro codes. Thus successfully solving them shows the robustness of the proposed algorithm.

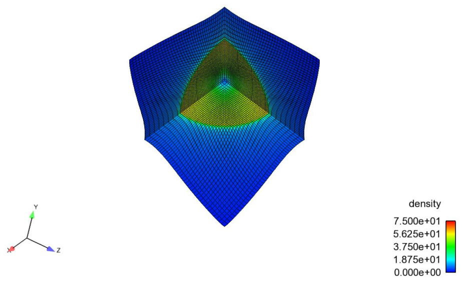
Noh

The Noh Problem describes a gamma-law gas impacting the origin with a velocity of $1\frac{cm}{\mu s}$.

Density



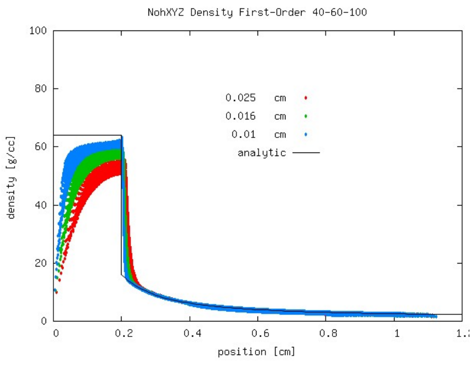
(a) time $t = 0\ \mu s$



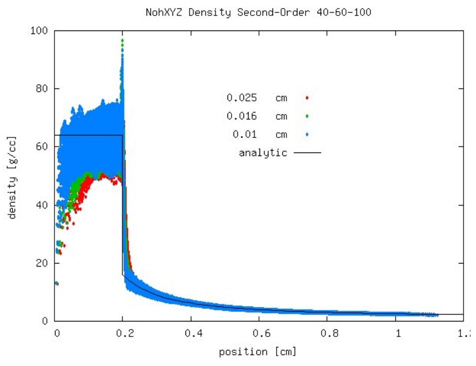
(b) time $t = 0.6\ \mu s$

Figure 115

Figure 115a shows the undeformed block of gas for the Noh problem at time $t = 0$, the mesh is a 40x40x40 grid structure with equal cells. It is colored by the density of the gas in the mesh and is initially uniform. Figure 115b shows the Noh problem at 0.6 microseconds. Here we can see the deformation of the entire mesh and a strong increase in density toward the origin, there is a slight numerical error in the cell closest to the origin. It can be noted that the mesh is symmetric across each axis.



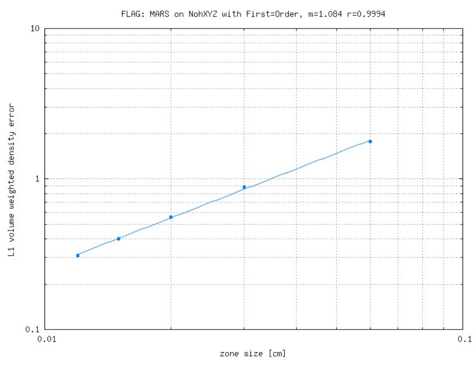
(a) First Order



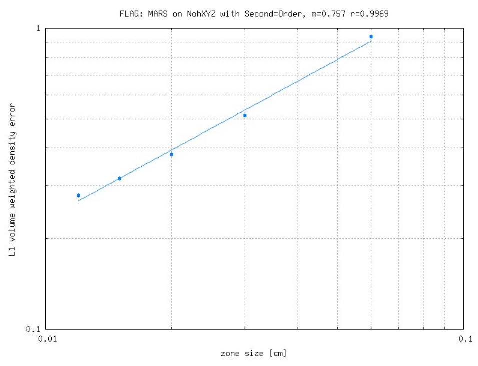
(b) Second Order

Figure 116

Figure 116a shows both the analytical solution and the numerical solution for mesh sizes 40, 60, and 100 with the first order approximation on the Noh problem. As the mesh size increases it is obvious to see that the analytical solution approaches the numerical solution. Because the second order case succeeded for all mesh sizes on the Noh problem without corner pressures, it was selected as the next plot to show. It can be noted here that after $x = 0.2\ cm$ the second order solution is much closer to the approximate than the first order solution. Figure 116b depicts the second order case. It is noticeable that the second order approximation is highly oscillatory for x less than $0.2cm$ (i.e. behind the shock). Future work may find it helpful to investigate improved limiters that will more readily transition to the first order solution behind the shock.



(a) First Order Convergence

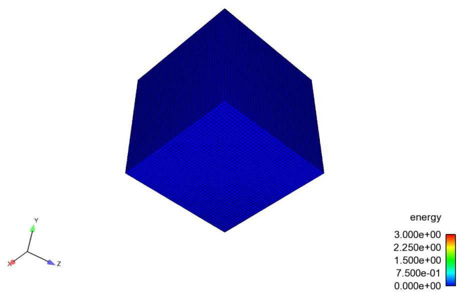


(b) Second Order Convergence

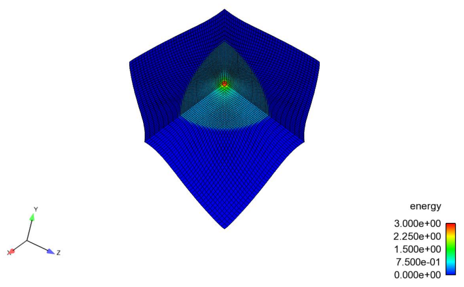
Figure 117

Figures 117a and 117b illustrate the rates of convergence for the first and second order approximations. In these figures m is the slope of convergence and r is the quality of fit for the least squares method. it is somewhat surprising to see that figure 117a has a better fit and a better rate of convergence than figure 117b. This might suggest that the first order method works better for this problem. However, referring back to figures 116a and 116b it can be deduced that the first order solution is simply less oscillatory and more precise but the second order solution is a more accurate fit to the exact solution. This does suggest that for this case a very high resolution first order approximation will yield the best results given that calculation time cost is not a concern.

Energy



(a) time $t = 0\ \mu s$

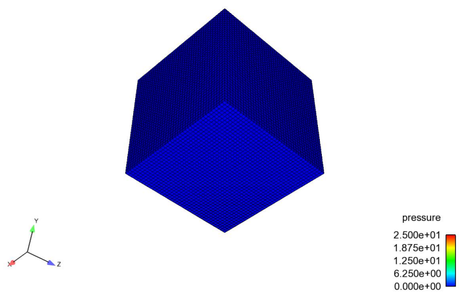


(b) time $t = 0.6\ \mu s$

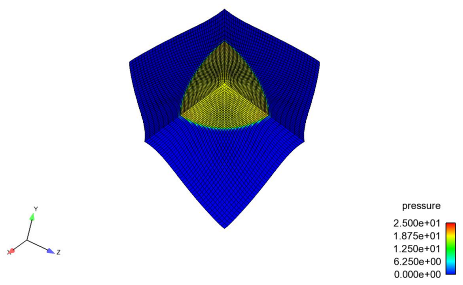
Figure 118

Figures 118a and 118b show the internal energy for the first order 40x40x40 mesh in the Noh problem.

Pressure



(a) time $t = 0\ \mu s$



(b) time $t = 0.6\ \mu s$

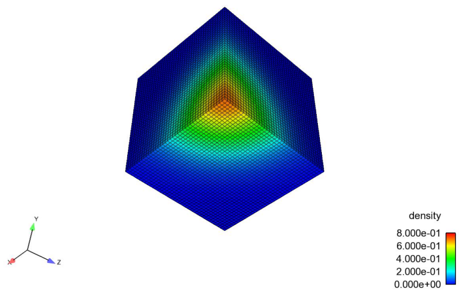
Figure 119

Figures 119a and 119b show the pressure for the first order 40x40x40 mesh in the Noh problem.

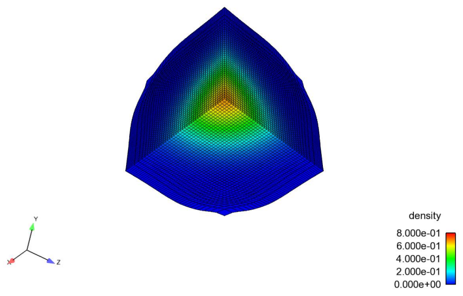
Kidder

The Kidder problem describes a shock free compression and expansion of a gamma-law ball of gas.

Density



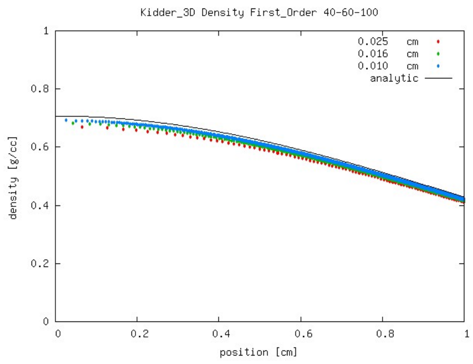
(a) time $t = 0\ \mu s$



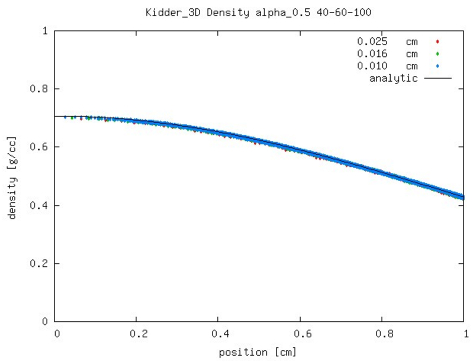
(b) time $t = 2.0\ \mu s$

Figure 120

Figure 120a shows the undeformed block of gas for the Kidder problem at time $t = 0$, the mesh is a 40x40x40 grid structure with equal cells. It is colored by the density of the gas in the mesh and is initially uniform. Figure 120b shows the Kidder problem at 2.0 microseconds.



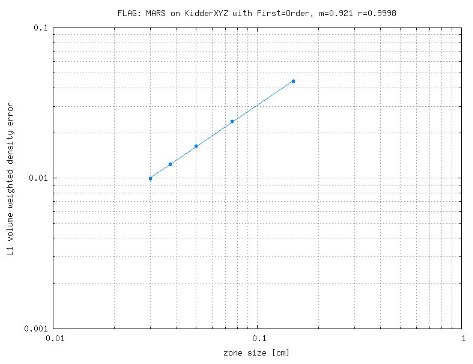
(a) First Order



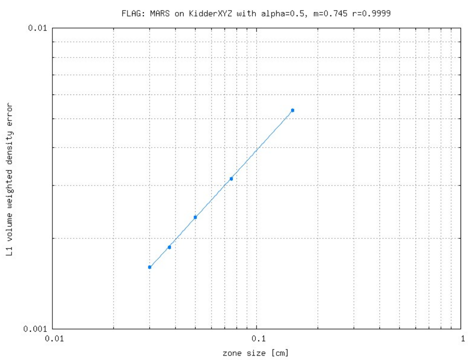
(b) $\alpha = 0.5$

Figure 121

Figure 121a shows both the analytical solution and the numerical solution for mesh resolutions 40, 60, and 100 with the first order approximation on the Kidder problem. As the mesh size increases it is obvious to see that the analytical solution approaches the numerical solution in a fashion very similar to the Noh Problem. The second order case without corner pressures did not reach the completion time for all mesh sizes on the Kidder problem due to mesh tangling. Only the two coarsest mesh resolutions worked with second order and without corner pressures. The first case to have success on all tested mesh sizes was with the parameter $\alpha = 0.5$, thus it was selected as the plot to compare the first order solution against. For this case the second order solution is much closer to the exact solution than the first order solution. Figure 121b depicts the second order case. Both the second with corner pressures and first order approximations appear to be very stable here. We do acknowledge, though, that numerical errors are present after x = 1 cm that prevent well behaved convergence. Thus scatter plots for this method were only included up to this point.



(a) First Order



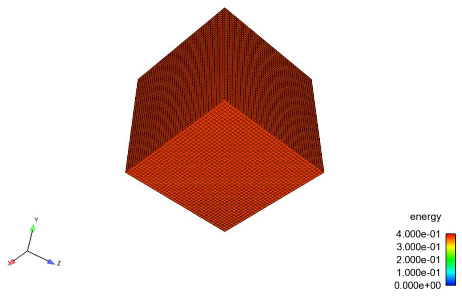
(b) $\alpha = 0.5$

Figure 122

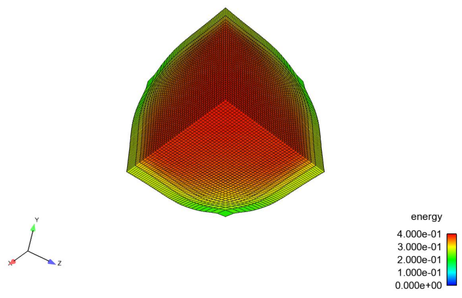
Figures 122a and 122b illustrate the rates of convergence on the Kidder problem for the first order approximation and the second order approximation with parameter $\alpha = 0.5$. In these figures m is the slope of convergence and r is the quality of fit for the least squares method.

Figure 122a has a better rate of convergence than figure 122b, this is likely because the lower mesh resolutions are further from the exact solution in the first order case compared to the second order cases. This idea is somewhat confirmed by noting that the least squares method fit is almost the same in both cases.

Energy



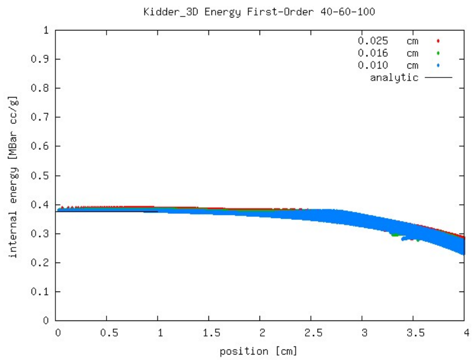
(a) time $t = 0$ μs



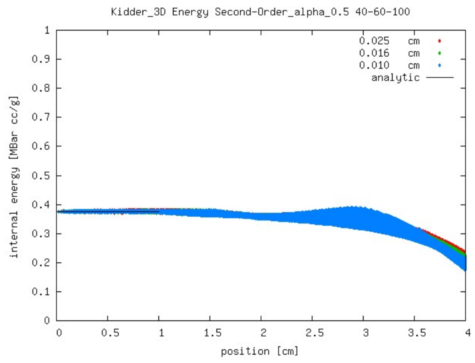
(b) time $t = 2.0$ μs

Figure 123

Figure 123a shows the undeformed block of gas for the Kidder problem at time $t = 0$, the mesh is a 40x40x40 grid structure with equal cells. It is colored by the energy of the gas in the mesh and is initially uniform. Figure 123b shows the Kidder problem at 2.0 microseconds.



(a) First Order

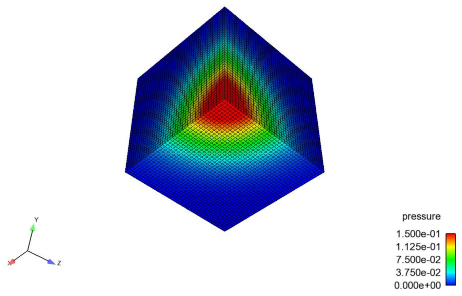


(b) $\alpha = 0.5$

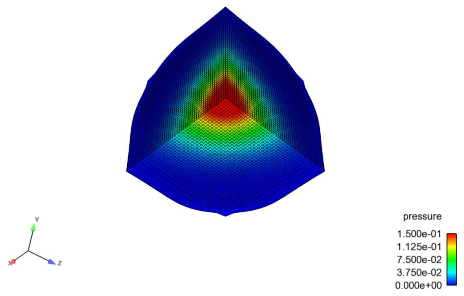
Figure 124

Figure 124a shows both the analytical solution and the numerical solution for mesh resolutions 40, 60, and 100 with the first order approximation on the Kidder problem. As the mesh resolution increases it is obvious to see that the analytical solution approaches the numerical solution in a fashion very similar to the Noh Problem. The second order case without corner pressures did not succeed for all mesh sizes on the Kidder problem. The first case to have success on all tested mesh sizes was with the parameter $\alpha = 0.5$, thus it was selected as the plot to compare the first order solution against. For this case the second order solution with corner pressures is much closer to the exact solution than the first order solution until $t = 1$, after which the second order solution with corner pressures tends to oscillate a bit more. Figure 124b depicts the second order case. The first order approximation appears to be more stable for long times.

Pressure



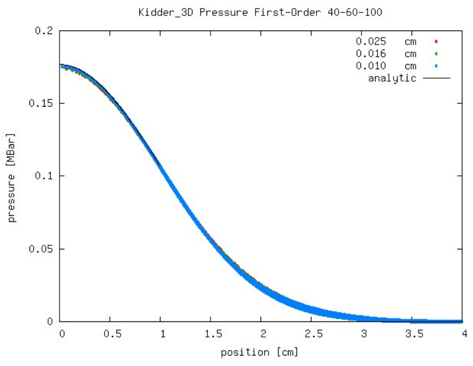
(a) time $t = 0$ μs



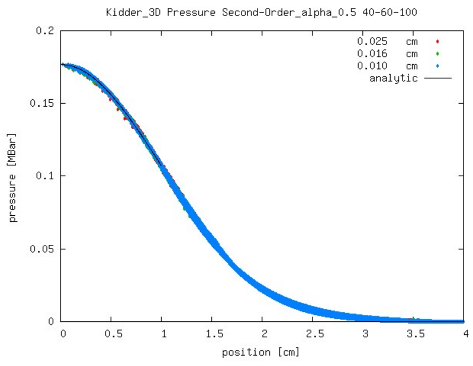
(b) time $t = 2.0$ μs

Figure 125

Figure 125a shows the undeformed block of gas for the Kidder problem at time $t = 0$, the mesh is a 40x40x40 grid structure with equal cells. It is colored by the pressure of the gas in the mesh and is initially uniform. Figure 125b shows the Kidder problem at 20 time steps or 2.0 seconds later. The pressure becomes more concentrated at $t = 2.0$ microseconds than it was at $t = 0$



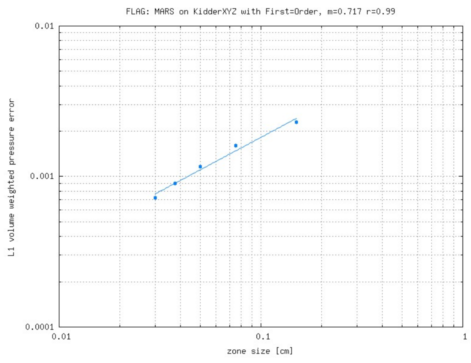
(a) First Order



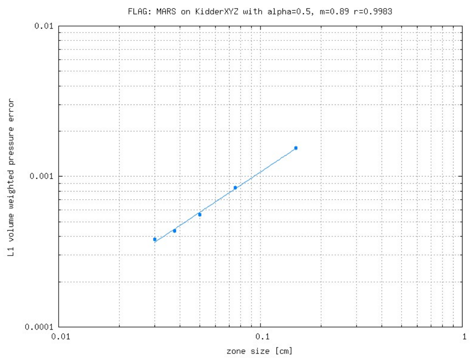
(b) $\alpha = 0.5$

Figure 126

Figure 126a shows both the analytical solution and the numerical solution for mesh resolutions 40, 60, and 100 with the first order approximation on the Kidder problem. As the mesh size increases it is obvious to see that the analytical solution approaches the numerical solution in a fashion very similar to the Noh Problem. The second order case without corner pressures did not succeed for all mesh sizes on the Kidder problem. The first case to have success on all tested mesh sizes was with the parameter $\alpha = 0.5$, thus it was selected as the plot to compare the first order solution against. As has been the usual the second order solution with corner pressures is much closer to the exact solution than the first order solution. Figure 126b depicts the second order case. The second order case appears to be more stable for pressure than for energy.



(a) First Order



(b) $\alpha = 0.5$

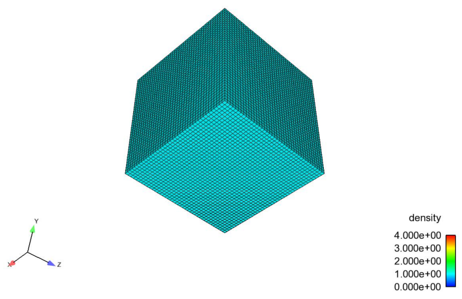
Figure 127

Figures 127a and 127b illustrate the rates of convergence on the Kidder problem for the first order approximation and the second order approximation with parameter $\alpha = 0.5$. In these figures m is the slope of convergence and r is the quality of fit for the least squares method. In this case the second order method has clearly a better fit. Figure 127b has a better rate of convergence and a better least squares fit than figure 127a.

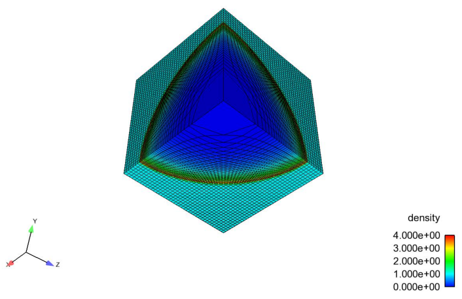
Sedov

The Sedov problem describes a gamma-law gas with a blast wave generated by an energy source at the origin

Density



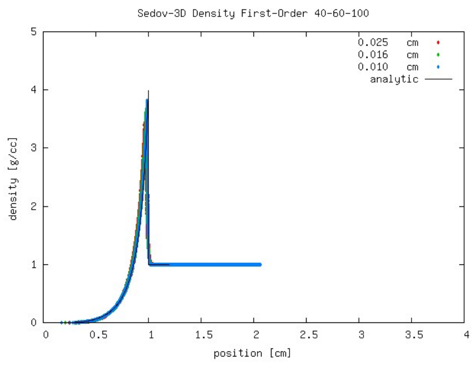
(a) time $t = 0\ \mu s$



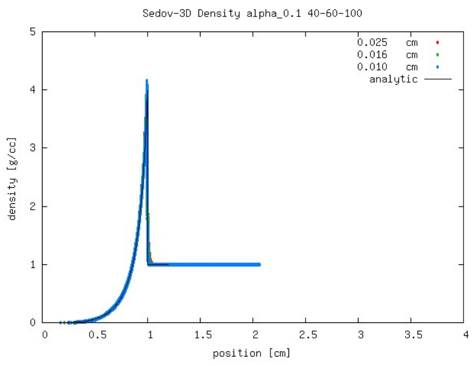
(b) time $t = 1.0\ \mu s$

Figure 128

Figure 128a shows the undeformed block of gas for the Sedov problem at time $t = 0$, the mesh is a 40x40x40 grid structure with equal cells. The mesh is colored by the density of the gas in the mesh and is initially uniform. Figure 128b shows the Sedov problem at 1.0 microsecond.



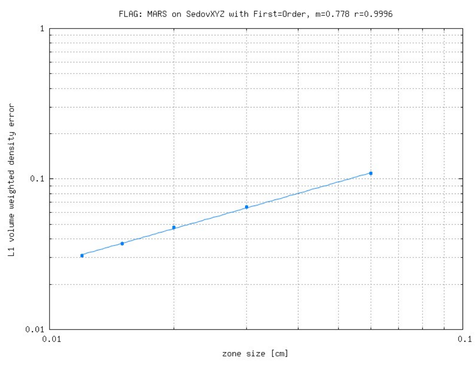
(a) First Order



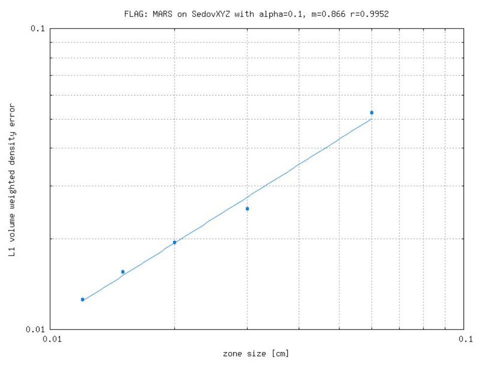
(b) $\alpha = 0.1$

Figure 129

Figure 129a shows both the analytical solution and the numerical solution for mesh resolutions 40, 60, and 100 with the first order approximation on the Sedov problem. As the mesh size increases it is obvious to see that the analytical solution approaches the numerical solution in a fashion very similar to the Noh Problem. The second order case without corner pressures did not succeed for all mesh sizes on the Sedov problem. The first case to have success on all tested mesh sizes was with the parameter $\alpha = 0.1$, thus it was selected as the plot to compare the first order solution against. For this case the second order solution with corner pressures is much closer to the exact solution than the first order solution. Figure 129b depicts the second order case. Both the second and first order approximations appear to be very stable here.



(a) First Order

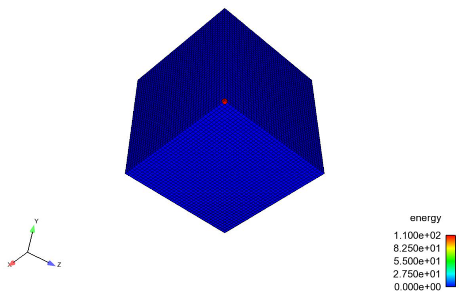


(b) $\alpha = 0.1$

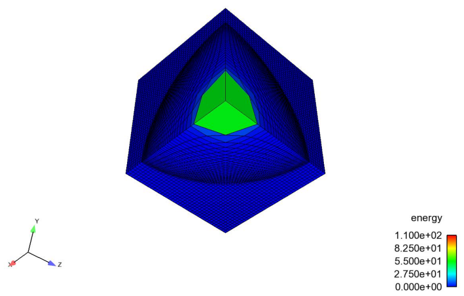
Figure 130

Figures 130a and 130b illustrate the rates of convergence on the Sedov problem for the first order approximation and the second order approximation with parameter $\alpha = 0.1$. In these figures m is the slope of convergence and r is the quality of fit for the least squares method. In this case the second order method is clearly a better method. Figure 130b has a better rate of convergence and a slightly worse least squares fit than figure 130a.

Energy



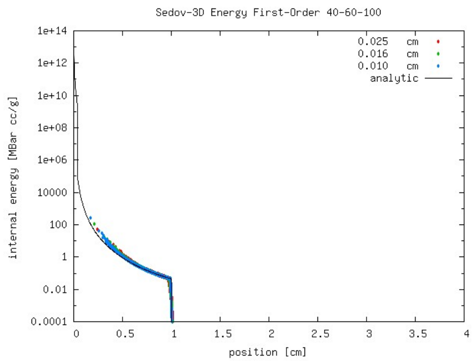
(a) time $t = 0$ μsec



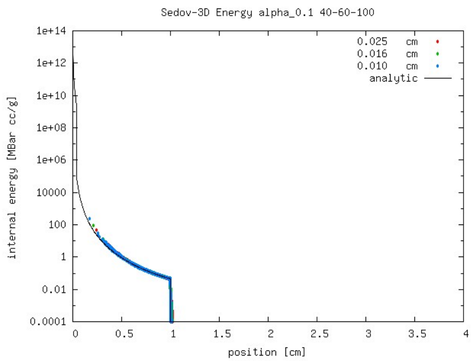
(b) time $t = 1.0$ μsec

Figure 131

Figure 131a shows the undeformed block of gas for the Sedov problem at time $t = 0$, the mesh is a 40x40x40 grid structure with equal cells. The mesh is colored by the specific internal energy of the gas. Figure 131b shows the Sedov problem at 10 time steps or 1.0 seconds later. It can be seen that the initial time step is not uniform but rather a large amount of specific internal energy is contained in the cell closest to the origin. In the latter Figure, the outward diverging shock wave can be seen traveling through the cube.



(a) First Order

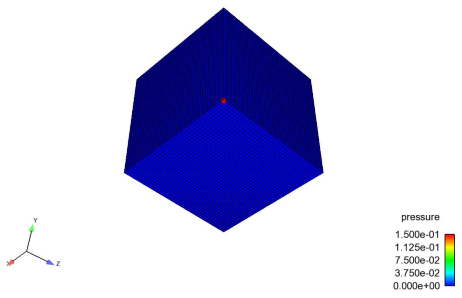


(b) $\alpha = 0.1$

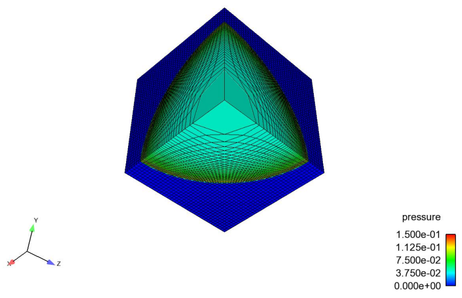
Figure 132

Figure 132a shows both the analytical solution and the numerical solution for mesh resolutions 40, 60, and 100 with the first order approximation on the Sedov problem. As the mesh size increases it is obvious to see that the analytical solution approaches the numerical solution in a fashion very similar to the Noh Problem. The second order case without corner pressures did not succeed for all mesh sizes on the Sedov problem. The first case to have success on all tested mesh sizes was with the parameter $\alpha = 0.1$, thus it was selected as the plot to compare the first order solution against. For this case the second order solution is much closer to the exact solution than the first order solution. Figure 132b depicts the second order case. Both the second and first order approximations appear to be very stable here.

Pressure



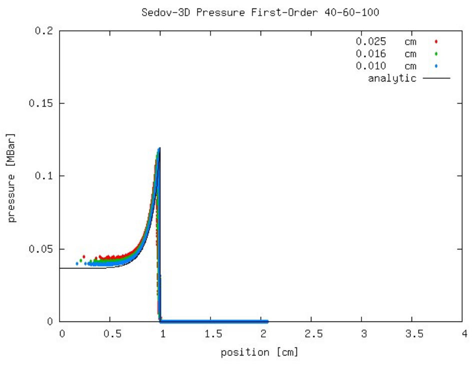
(a) time $t = 0\ \mu s$



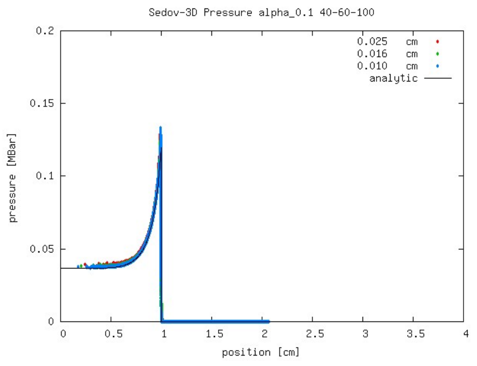
(b) time $t = 1.0\ \mu s$

Figure 133

Figure 133a shows the undeformed block of gas for the Sedov problem at time $t = 0$, the mesh is a 40x40x40 grid structure with equal cells. The mesh is colored by the pressure of the gas. Figure 133b shows the Sedov problem at 1.0 microseconds. It can be seen that at time equals 0 a large amount of pressure is contained in the cell closest to the origin. In the latter Figure, the outward traveling diverging wave can be seen traveling through the cube.



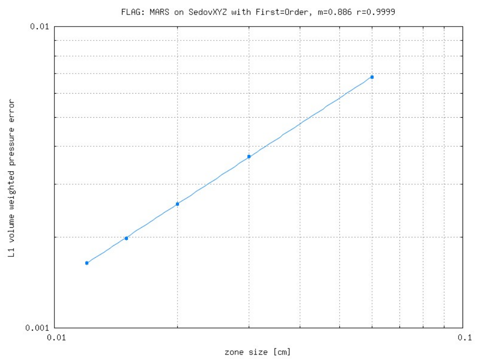
(a) First Order



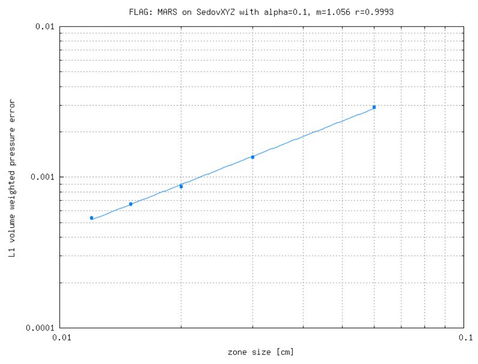
(b) $\alpha = 0.1$

Figure 134

Figure 134a shows both the analytical solution and the numerical solution for mesh resolutions 40, 60, and 100 with the first order approximation on the Sedov problem. As the mesh size increases it is obvious to see that the analytical solution approaches the numerical solution in a fashion very similar to the Noh Problem. The second order case without corner pressures did not succeed for all mesh sizes on the Sedov problem. The first case to have success on all tested mesh sizes was with the parameter $\alpha = 0.1$, thus it was selected as the plot to compare the first order solution against. For this case the second order solution with corner pressures is much closer to the exact solution than the first order solution. Figure 134b depicts the second order case. Both the second and first order approximations appear to be very stable here.



(a) First Order



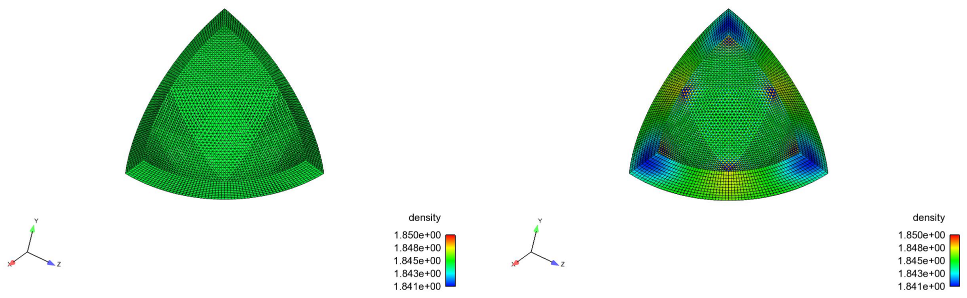
(b) $\alpha = 0.1$

Figure 135

Figures 130a and 130b illustrate the rates of convergence on the Sedov problem for the first order approximation and the second order approximation with parameter $\alpha = 0.1$. In these figures m is the slope of convergence and r is the quality of fit for the least squares method. In this case the second order method is clearly a better method. Figure 130b has a better rate of convergence and a slightly worse least squares fit than figure 130a.

Verney

Density

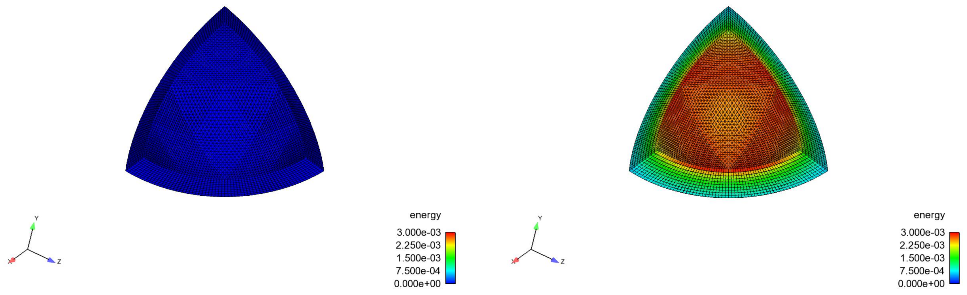


(a) time $t = 0\ \mu s$ (b) time $t = 130\ \mu s$

Figure 136

Figure 136a shows the initial setup of the Verney problem (at time $t = 0$). The mesh is colored by the density of the gas in the mesh and is initially uniform. Figure 136b shows the Verney problem at 130 microseconds. The inner radius of the calculation on the three coordinate axes at 130 microseconds is 3.59 cm where the exact solution is 3.0. Future work should use finer radial resolutions to assess convergence rate.

Energy

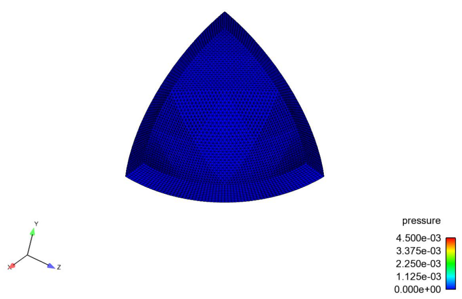


(a) time $t = 0\ \mu s$ (b) time $t = 130\ \mu s$

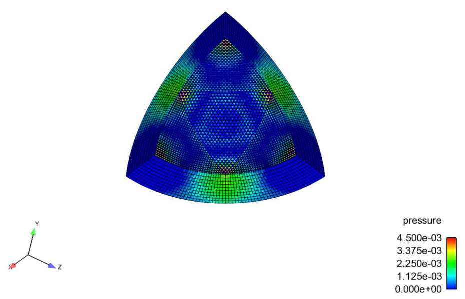
Figure 137

Figure 137a shows the initial setup of the Verney problem (at time $t = 0$). The mesh is colored by the specific internal energy of the gas. Figure 137b shows the Verney problem at 130 microseconds.

Pressure



(a) time $t = 0\ \mu s$



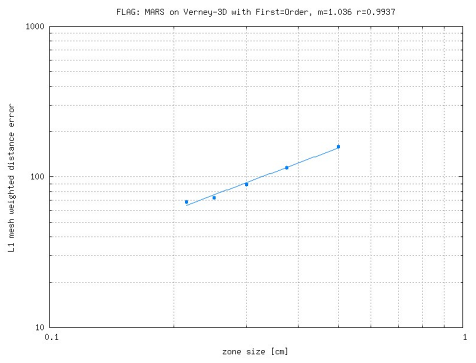
(b) time $t = 130\ \mu s$

Figure 138

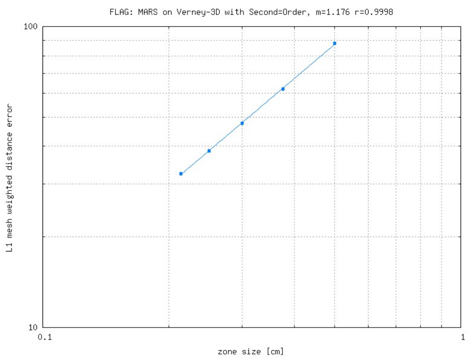
Figure 138a shows the initial setup of the Verney problem (at time $t = 0$). The mesh is colored by the pressure of the gas in the mesh and is initially uniform. Figure 138b shows the Verney problem at 130 microseconds.

Distance

below are L1 error plots for the distance the material has deformed along the X, Y, and Z axes. Convergence for this problem was tested with 6 refinements on the inner surface and 6,8,10,12,and 14 radial zones for each mesh generation.

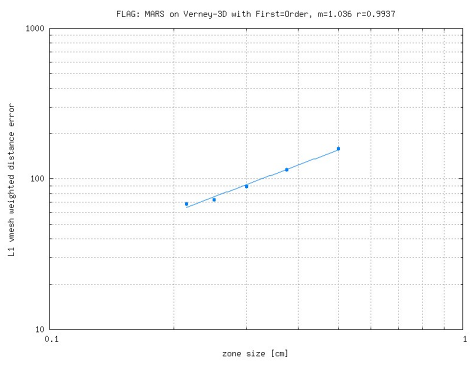


(a) X First Order

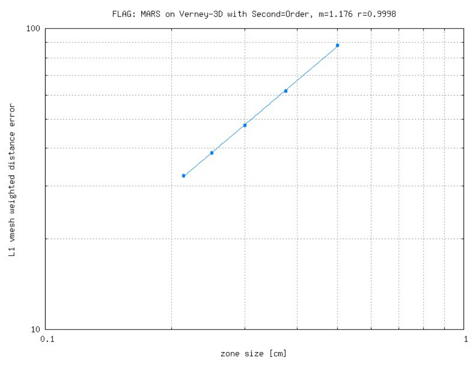


(b) X Second Order

Figure 139

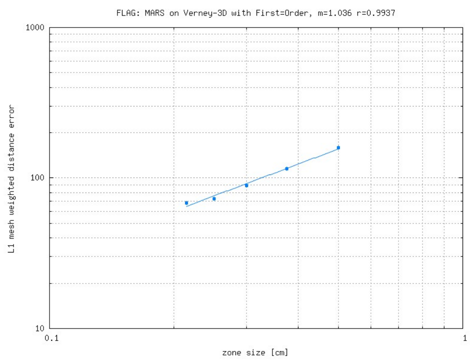


(a) Y First Order

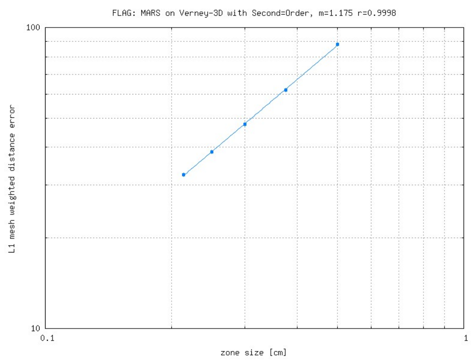


(b) Y Second Order

Figure 140



(a) Z First Order



(b) Z Second Order

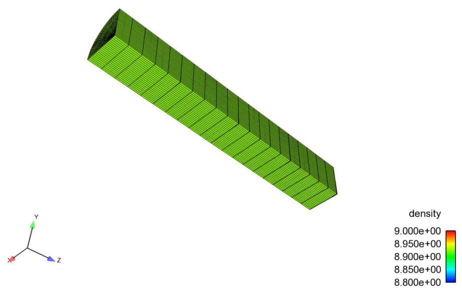
Figure 141

All cases show a good least squares fit, however, the second order cases show an improved least squares fit and have a greater order of convergence. whereas the first order cases only have a first order rater of convergence.

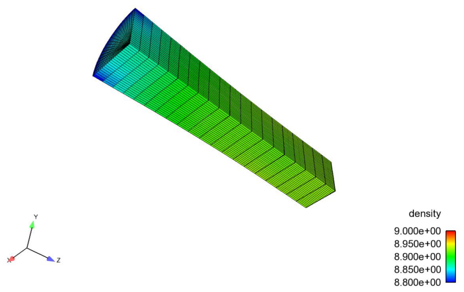
Taylor Anvil

Density

The Taylor Anvil problem describes a rod plasticly deforming after impacting a wall. Below are the initial and final images (figures 142a and 142b respectively)for the problem colored by density.



(a) time $t = 0\ \mu s$

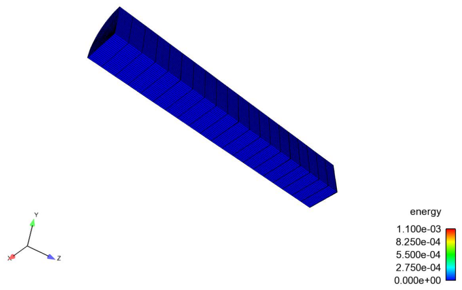


(b) time $t = 150\ \mu s$

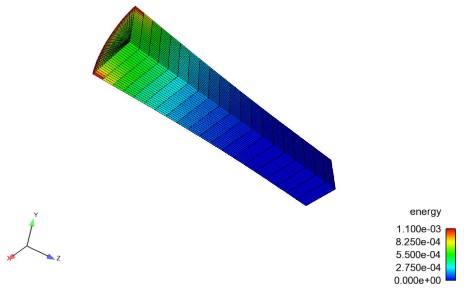
Figure 142

Energy

Below are the initial and final images (figures 143a and 143b respectively)for the problem colored by energy.



(a) time $t = 0\ \mu s$

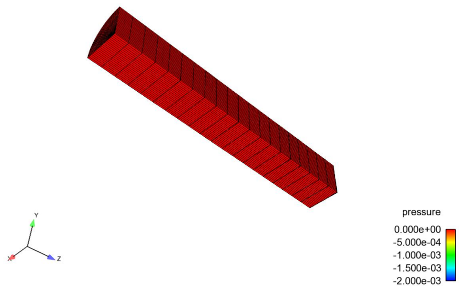


(b) time $t = 150\ \mu s$

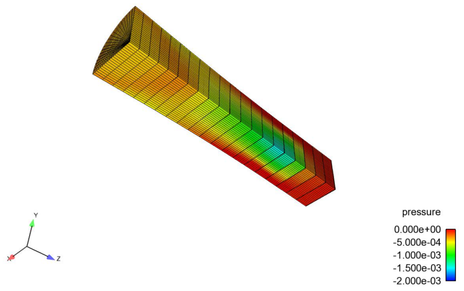
Figure 143

pressure

Below are the initial and final images (figures 144a and 144b respectively)for the problem colored by pressure.



(a) time $t = 0\ \mu s$



(b) time $t = 150\ \mu s$

Figure 144

Conclusions

In this paper, a 3D Lagrangian staggered grid Godunov-like approach was evaluated. The specific Godunov approach used here uses a Multi-directional Approximate Riemann Solution (MARS). A set of five test problems were used to assess the accuracy and robustness of both the first order and second order accurate approaches. These problems included the Noh, Sedov, Kidder, Verney, and Taylor Anvil. Analytical solutions to the test problems were compared to numerical solutions where possible. Solutions improved in accuracy with mesh refinement. It has also been noted throughout the test problems that numerical oscillations can arise with the second order method. In contrast, the first order method generates very smooth symmetric solutions. In either case both methods show that the 3D MARS method will converge to the exact solution.

Verification Assessment of Initializaton Strategies for Shock Test Problem Simulations

By Philip Ivancic and Jennifer Lilieholm, Scott Ramsey (Mentor)

Introduction

In the past few decades, computer simulations have aided in the engineering production process of designing, prototyping, testing, and redesigning. Prototyping can be very expensive, and the process could have to be cycled through several times before an optimized product can be produced. Computer simulations aid in this process by predicting the testing before a prototype needs to be made, which greatly reduces the cost of the design process, or in situations where a physical experiment can not be accurately or safely conducted. However, it is important to justify the confidence in the computational solutions through verification and validation. In short, verification is the process of showing that the code is solving the numerical model consistent with the programmer's expectations while validation is showing how well the model predicts reality [48].

One way to verify the code is to compare the results of simulations to an analytical solution through a mesh refinement study. It is recommended [Scott Ramsey, personal communication, 2014] that at least four different resolutions are used in the refinement study. Each refined solution can be plotted against the analytical, creating a flow plot. As the mesh gets more refined, the computational solutions should converge to the analytical. If they do not, then that is a good indication that there might be something wrong with the code, but it could also be a problem with the boundary conditions and that the "verifier" is not simulating the same problem. Some people would be tempted to stop here for the verification, but this is neither quantitative nor rigorous. The flow plot is a qualitative metric, but a quantitative metric is much more desirable for verification. The error for each solution can be calculated by

$$error_i = |solution_i - exact_i|, wherei = gridindex \tag{169}$$

and plotted. It can now be seen how the error, itself, changes among the different resolutions. Another important quantity to calculate, though, for verification would be the convergence rate. The error at each resolution can be normalized by using a L1 norm

$$Norm = \frac{\sum(w_i \times error_i)}{\sum w_i} \tag{170}$$

where w is some weighting parameter. These norms are plotted against the spatial discretization, Δx. A power law fit could be found for that data in the form of

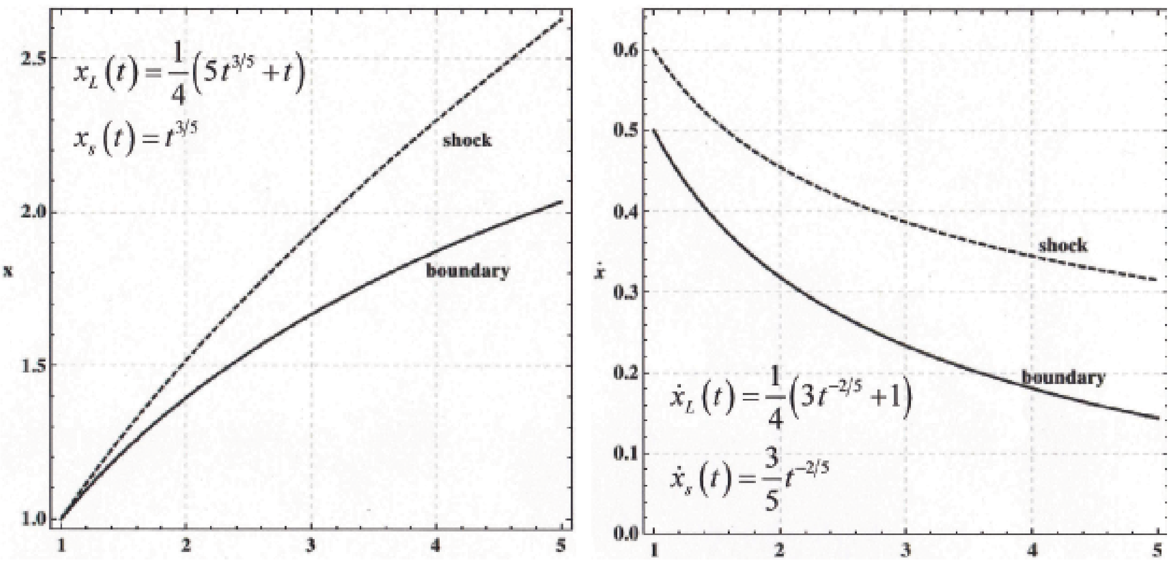
$$L1 = A * \Delta x^B \tag{171}$$

where A is the convergence constant and B is the convergence rate. The convergence constant is a number related to the error; the lower this constant the better. The convergence rate is related to the order of accuracy of the numerical method implemented, and they should match up, though usually not exactly. This provides the best evidence whether or not the numerical method is correctly implemented in the code.

One thing to consider when doing the verification is that the test problems to be simulated are sometimes defined over all space and for all time. Computer simulations are in a bounded space and time, so a decision has to be made about how to set up the initial and boundary conditions for the problem. Previous work has been done by Ramsey [54] describing two ways to set up these conditions for compressible, hydrodynamics test problems involving shocks in a Lagrangian mesh, which will also be used in this paper: direct method and piston method.

The direct method has the shock initialized in the computational domain from the beginning. This region is subdivided into a pre-shock and a post-shock region. Within each of these regions, the initial conditions are set based on the analytical solution, which would be known, for the chosen initialization time. The variables that would have to be initialized are the density, internal energy, and velocity. The velocity would also have to be set in the boundary conditions. As the time progresses during the simulation from the initial time, the shock will move along the domain.

The piston method is analogous to a shock being caused by a piston [35]. As a piston compresses a fluid, a shock will form and move ahead of the piston. In this method one of the boundaries will act as the piston head that moves at the post-shock fluid velocity. The shock position at any time is known via the analytical solution, so an initial time is chosen with the piston and shock coincident. Figure 145 shows an example of the piston and shock positions and velocities [54]. In this setup, since the shock starts off at one of the boundaries, the initial conditions for the domain are only the pre-shock quantities. Again, for this method, the boundary conditions would be the velocities, with the "piston" boundary having the post-shock velocity.



(a) Position time history. (b) Velocity time history.

Figure 145: Piston setup example for Zel'dovich problem.

Strong Shock Assumption

The one-dimensional, inviscid hydrodynamic flow equations were solved with a calorically perfect gas, which assumptions that the specific heats are constant. With these assumptions, Coggeshall [11] derived the continuity, momentum, and energy equations to be

$$\begin{aligned} \rho_t + u\rho_r + \rho u_r + k\rho u/r &= 0 \\ u_t + uu_r + (1/\rho)\Gamma T\rho_r + \Gamma T_r &= 0 \\ \frac{\Gamma}{\gamma-1}(T_t + uT_r) + \Gamma T u_r + \Gamma T\frac{ku}{r} &= 0 \end{aligned} \tag{172}$$

where k, Γ, and γ are the geometric factor, gas constant, and adiabatic exponent, respectively. Also, ρ, p, T, and u are the fluid's density, pressure, temperature, and velocity, respectively and will be consistent throughout the paper. k can be set to 0, 1, or 2 corresponding to planar, cylindrical, or spherical geometries. Γ can take any positive value while γ has to be greater than 1.

Coggeshall derived 22 analytical solutions [11] to Equation 172. Some of these solutions involved shocks, which need additional equations to relate the upstream and downstream states. If the mass, momentum, and energy fluxes are continuous, then the connection between the upstream-downstream states, where the indexes 2 and 1 refer to upstream and downstream respectively, are determined by the Rankine-Hugoniot jump conditions [35] which lead to the following ratios:

$$\frac{\rho_2}{\rho_1} = \frac{(\gamma-1)p_1 + (\gamma+1)p_2}{(\gamma+1)p_1 + (\gamma-1)p_2}, \tag{173}$$

$$\frac{T_2}{T_1} = \frac{p_2}{p_1}\frac{(\gamma+1)p_1 + (\gamma-1)p_2}{(\gamma-1)p_1 + (\gamma+1)p_2}, \tag{174}$$

$$M_2{}^2 = \frac{2+(\gamma-1)M_1{}^2}{2\gamma M_1{}^2-(\gamma-1)} \tag{175}$$

where the Mach number, M, is related to the fluid velocity. The strong shock assumption says that $(\gamma-1)p_2$ is much larger than $(\gamma+1)p_1$. This means that the ratios $\frac{T_2}{T_1}$ and $\frac{p_2}{p_1}$ can be infinitely large and the density ratio $\frac{\rho_2}{\rho_1} = \frac{(\gamma+1)}{(\gamma-1)}$.

FLAG

Three test problems were simulated using FLAG [6, 9, 8]. FLAG is a Lagrangian hydrocode to conduct staggard grid hydrodynamics. It does have capabilities for rad-hydro, but that was not used in these test problems. Barton artificial viscosity was used to dampen post shock oscillations when simulating Eulerian equations. A polytropic equation of state was also implemented with the code.

Coggeshall 19

The equations given for the Coggeshall 19 (Cog19) Cartesian solutions are given by:

$$Post-Shock\,Equations:\ \rho(r,t)=\rho_0[(\gamma+1)/(\gamma-1)],\ u(r,t)=a,\ T(r,t)=u_0{}^2(\gamma-1)/2\Gamma \tag{176}$$

$$Pre-Shock\,Equations:\ \rho(r,t)=\rho_0,\ u(r,t)=u_0(<0)+a,\ T(r,t)=0 \tag{177}$$

$$Shock\,Location:\ R(t)=[-\frac{1}{2}(\gamma-1)u_0+a]t \tag{178}$$

This is essentially the Noh problem [47], which is a popular problem for verficiation. To add some variation to the solution, the equations were transformed with a Galilean boost in velocity. This velocity boost also allows the piston method to be used. If a is zero, then that would lead to the "piston" velocity also being zero, and, thus, the "piston" would not actually be compressing the flow.

One Dimensional Planar Results

Equations 176 - 178 were first simulated using the direct method. The initial domain size was 3 units long and discretized using 75, 150, 300, 600, and 1200 elements. The shock was initialized at 1 unit from the left boundary. The simulations ran for 1.3 simulated time units, at which point the shock was at 7.8 units.

Figure 146 shows the density flow viewgraph for each of the resolutions compared to the analytical at the end of the simulation. The first thing to notice is that the simulated shock is smeared behind the actual shock, so as the elements get smaller, the simulated shock becomes steeper and more accurately represents the actual shock. The other things are seen in the start up errors. Start up errors are caused by a discontinuity in the initial conditions [36], and shocks are discontinuities in solutions to Equation 172. It is noticed that the error on the left side gets smaller as the resolution increases, but the magnitude of the error in the middle does not change with resolution. However, as with the shock capturing, the error goes down because this error seems to be distributed over a certain number of elements. The error can be seen in Figure 147. The error magnitudes for the shock and error in the middle do not fall. However, the area underneath the curve goes down as the resolution goes up.

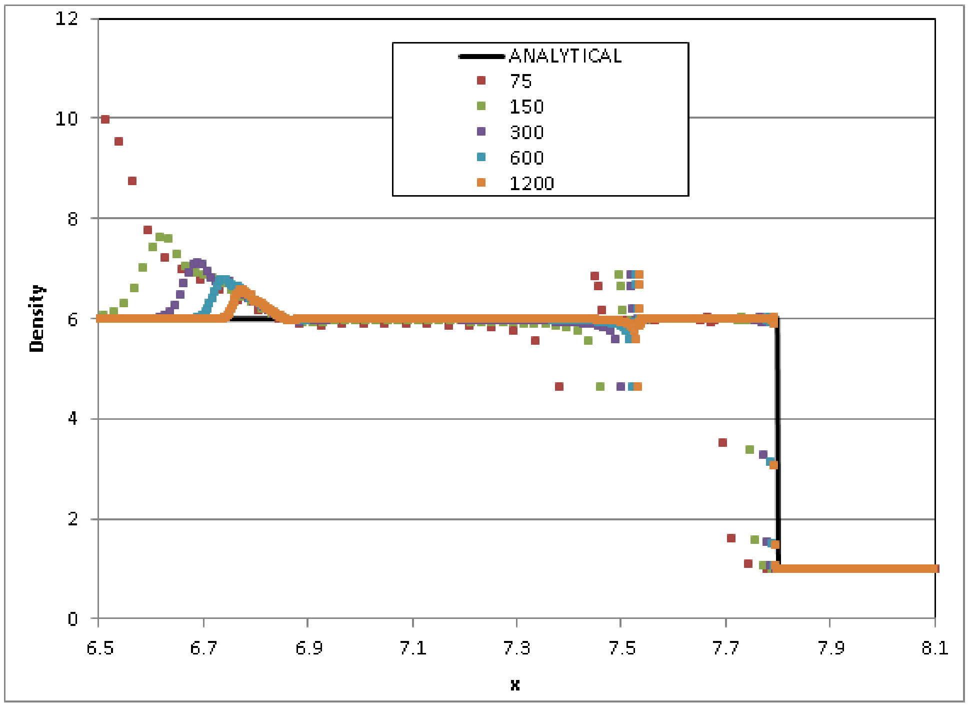


Figure 146: Coggeshall 19 direct method density flow viewgraphs.

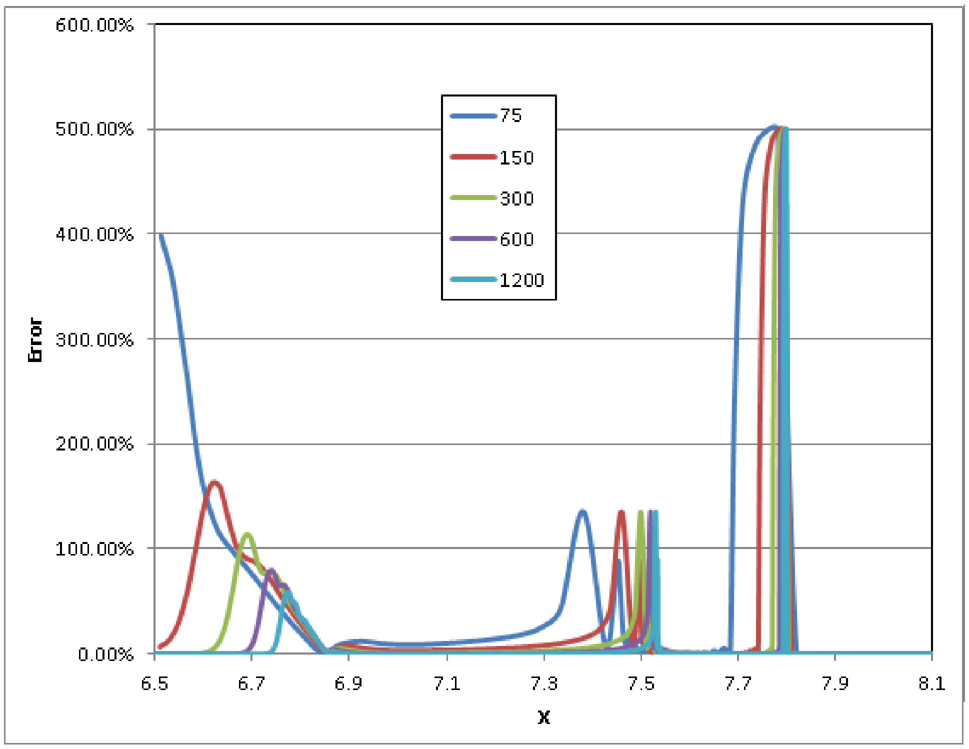


Figure 147: Coggeshall 19 direct method density error graphs.

The L1 error norms were calculated by using both a unweighted version and a volume-weighted version. The volume-weighted version can be expected to be the smaller of the two ways. The shock-capturing error and start up errors occur in the post-shocked region, and in this region, the element sizes could be decreased due to the compression caused by the shock. The norms were plotted, as seen in Figure 148, so that the convergence rate could be calculated. Both of the norms gave a convergence rate of around 1, which is what is expected from the FLAG code, or any code for that matter, for a shock. It should be noted that the 1200 element resolution was not used in the convergence plot since the piston method simulation was not run at that resolution. It would not be fair, when doing a comparison, to compare the convergence rates between the two methods if one method has more refinements than the other.

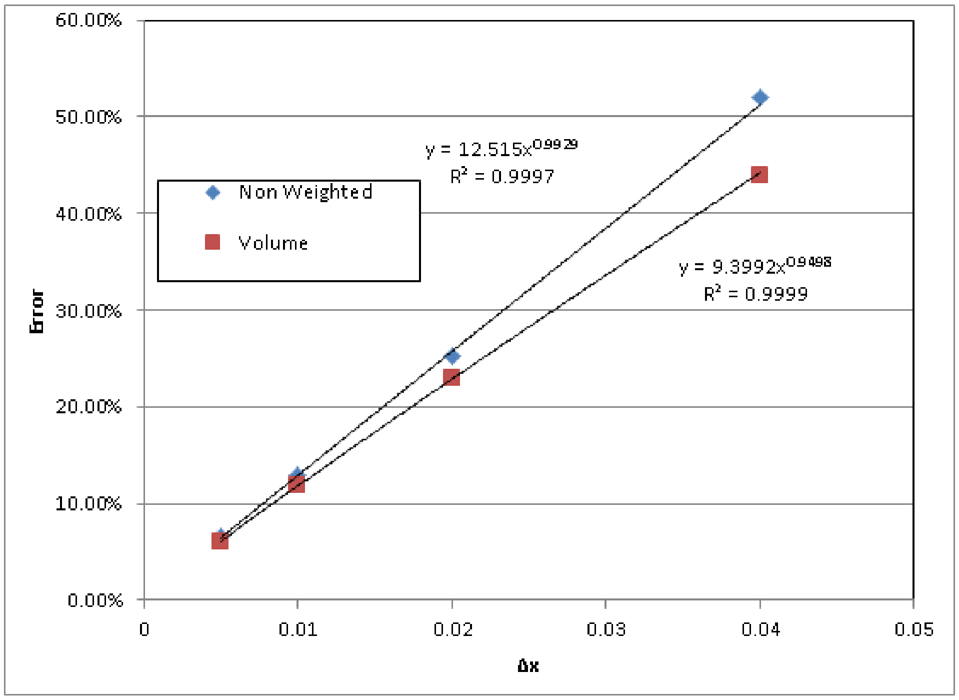


Figure 148: Coggeshall 19 direct method density L1 convergence. Unweighted and volume-weighted norms are both plotted.

The piston method simulation setup was essenially the same as the direct method. The domain was the same size and uses the same refinements (other than the finest one). However, due to the way it was set up, the shock started at 0 instead of 1, so the simulation had to run longer (1.5 simulated time) so that the shock would be in the same position as with the direct setup run.

Figure 149 shows the density flow viewgraph for the piston method setup run. One big difference in this method, opposed to the direct setup, is that the shock capturing is more smeared ahead of the shock, but still has more smearing behind the shock as well. Figure 150 shows a direct comparison of the two methods for the region around the shock for the coarsest resolution. It can be seen that the piston method does a better job at capturing the shock. The direct method smeared shock averages to a location of 7.720 and the piston method's shock averages to 7.812, which is closer to the analytical shock position of 7.8. The piston method also does not have a start up error associated with it since it does not have a discontinuity in the initialization, but it does have a wall heating error. The wall heating error in the piston setup acts the same as one of the start up errors in the fact that the magnitude does not change with resolution, as seen in Figure 151.

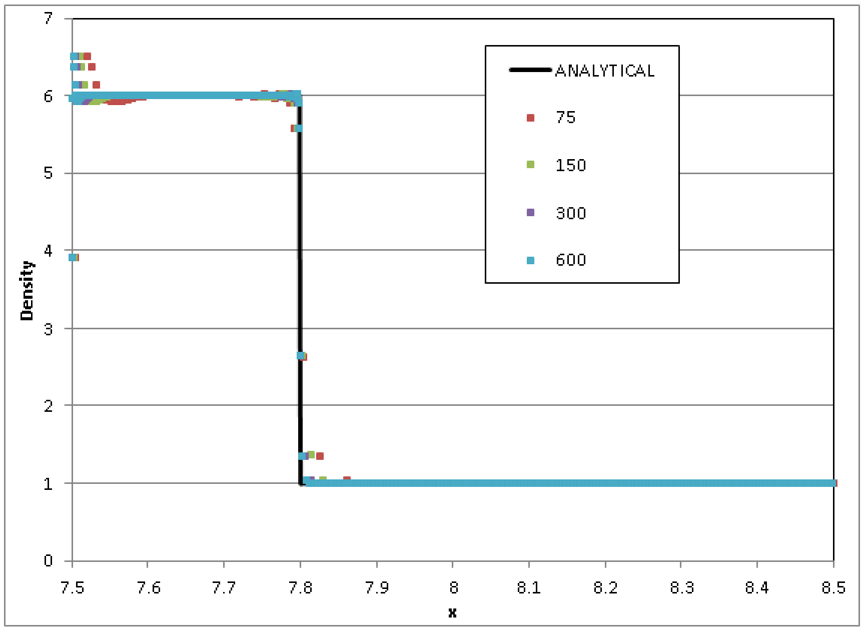


Figure 149: Coggeshall 19 piston method density flow viewgraph.

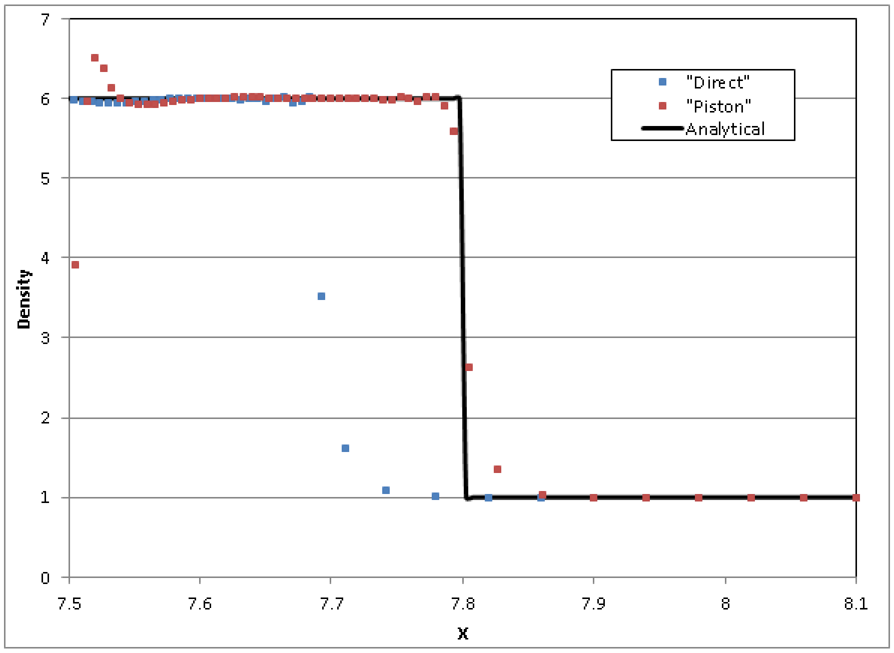


Figure 150: Coggeshall 19 direct and piston methods comparison. $\Delta x = 0.04$

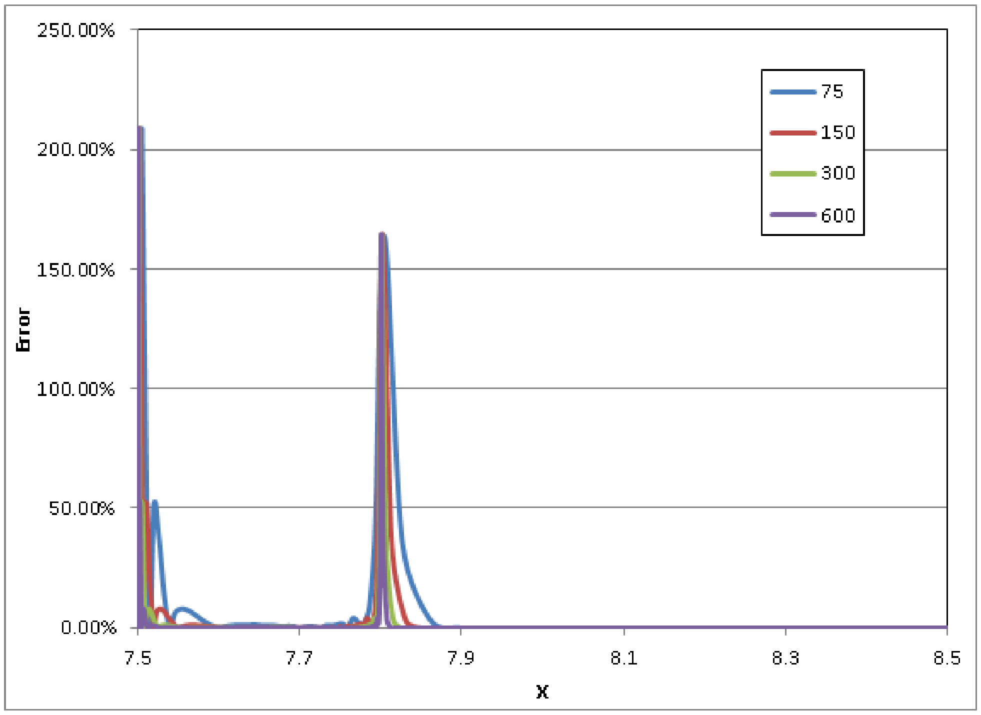


Figure 151: Coggeshall 19 piston method density error graphs.

Figure 152 shows the density convergence for the piston method. Again, it can be seen that the volume-weighted norms are less than that of the unweighted norms. The volume-weighted in the piston method is more exaggerated because the entire area behind the shock is compressed, so all of the errors are in a smaller volume cells. Again, the convergence rate is close to 1, but the piston method is ever so slightly closer than the direct method. This small difference does not give any meaningful information since the error asscioated with making the power-law curve fit to the limited data are likely to be more important than this miniscule differences. However, the convergence constant is much smaller for the piston method, which would mean that the errors associated with the piston method are smaller than the direct method. Tables 4 and 5 give a summary of the convergences between the two setup methods.

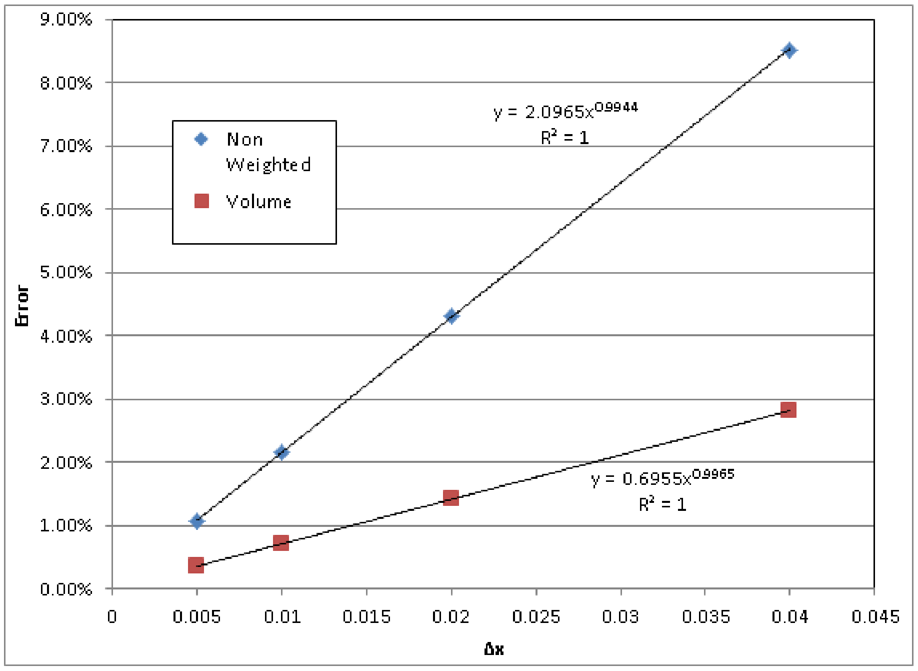


Figure 152: Coggeshall 19 piston method density L1 convergence. Unweighted and volume-weighted norms are both plotted.

Table 4: Coggeshall 19 one dimensional direct method convergence summary.

Direct Method			
	Weighting	Convergence Constant	Convergence Rate
Density	None	12.515	0.9929
	Volume	9.3992	0.9498
Pressure	None	3.5596	1.0441
	Volume	2.4994	0.9826
Temperature	None	0.4585	0.9663
	Volume	0.3953	0.9517
Velocity	None	0.6991	0.8956

Table 5: Coggeshall 19 one dimensional piston method convergence summary.

Piston Method			
	Weighting	Convergence Constant	Convergence Rate
Density	None	2.0965	0.9944
	Volume	0.6955	0.9965
Pressure	None	0.1708	1.0044
	Volume	0.0619	1.0025
Temperature	None	0.1442	0.9952
	Volume	0.0519	0.9972
Velocity	None	0.2501	0.9950

Further viewgraphs and convergence plots (for pressure, temperature, and velocity) can be seen in Appendix A of this paper.

Two Dimensional Planar Results

Verification in one dimension is a good place to start since it is generally easier to determine exact solutions to the one dimensional hydrodynamic equations. However, very few engineering problems are run in just one dimension; the more interesting problems are in higher dimensions. Verification needs to be done in these higher dimensions to show that the multi-dimensional numerical methods are accurately implemented. The Coggeshall 19 one dimensional solutions were extruded to a two dimensional domain. Essentially a quasi-one dimensional shock problem was simulated.

The one dimensional domain (3 units long) was used for the two dimensional case, but was 1 unit wide. The same four coarsest discretizations in the one dimensional simulation were also used for the x-coordinate discretizations, and the y-coordinate discretizations were chosen so that the undeformed mesh elements would be square. The two dimensional simulation also ran using both the direct and piston setup methods.

Since the two dimensional simulation was actually a quasi-one dimensional, the two dimensional results should collapse into the one dimensional results. The two dimensional results were cut parallel to the x-axis at y = 0.1, 0.25, and 0.5. All three of the slices had the same results, which is a good indication that there is not any variation along the y-axis and that this this is actually a quasi-one dimensional flow. The collapsed density solutions for the direct and piston methods on the coarsest mesh were then compared to the one dimensional solutions, as can be seen in Figures 153 and 154, respectively. Oddly, it appears that the two dimensional simulation with the direct method does better than the one dimensional. The start up errors are not as extreme in the two dimensional simulation and the simulated shock more accurately captures the analytical shock, which is at x = 7.8. The two dimension piston method collapsed almost to the one dimensional solution, but with a slightly more smeared shock.

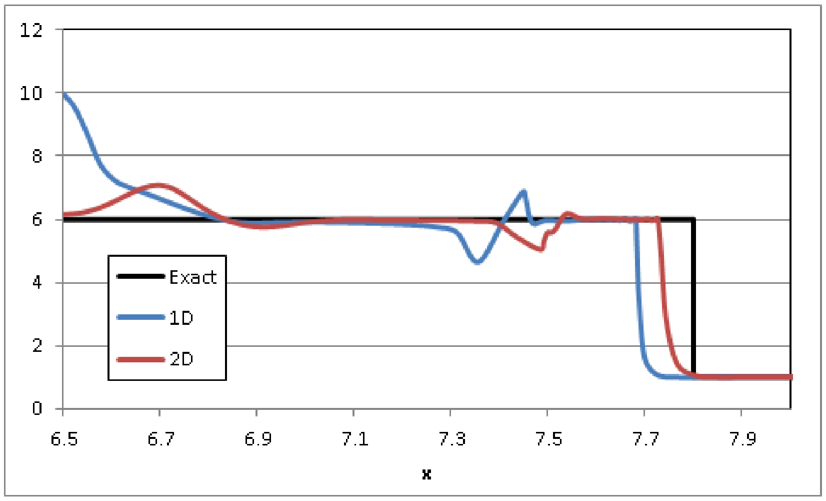


Figure 153: Coggeshall 19 direct method two dimensional collapse.

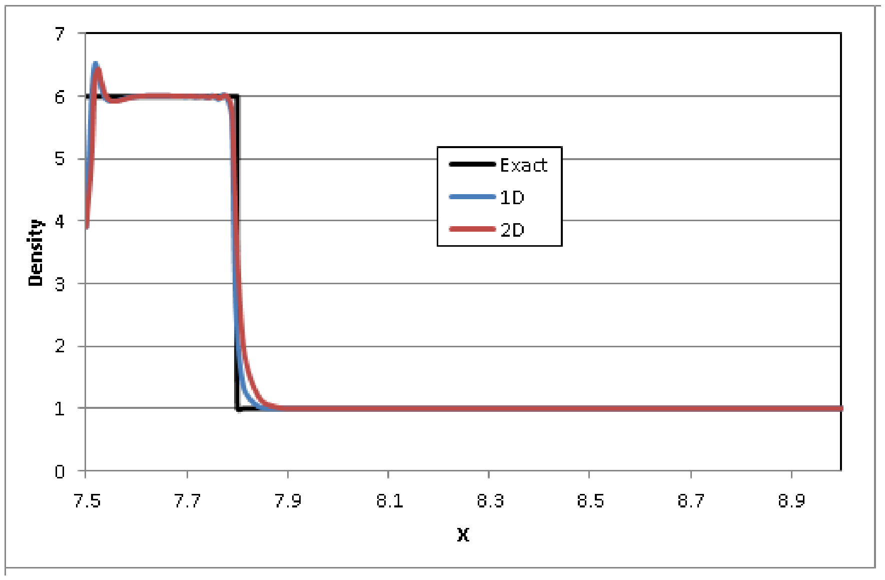


Figure 154: Coggeshall 19 piston method two dimensional collapse.

The convergence studies were performed for the two dimensional runs. The density convergences for the direct and piston methods can be seen in Figures 155 and 156, respectively. It should be noted that the volume-weighted norm was not used for the direct method. The method used to output the volume of the cells does so in decreasing volume order, but the locations are not outputted. Since with the direct method the areas vary on both sides of the shock due to the start up errors and the volumes can vary on the post-shocked side, research is currenlty undeway regarding how to correlate which side of the shock any given cell is on.

Again, for both methods, the errors converge at about first order as expected for a shocked simulation in FLAG. Another thing to notice is that the errors for the two dimension direct method are about half those for the one dimension simulation while the errors are about the same for poiston method simulations. Tables 6 and 7 give a summary of their convergences between the two setup methods in two dimensions.

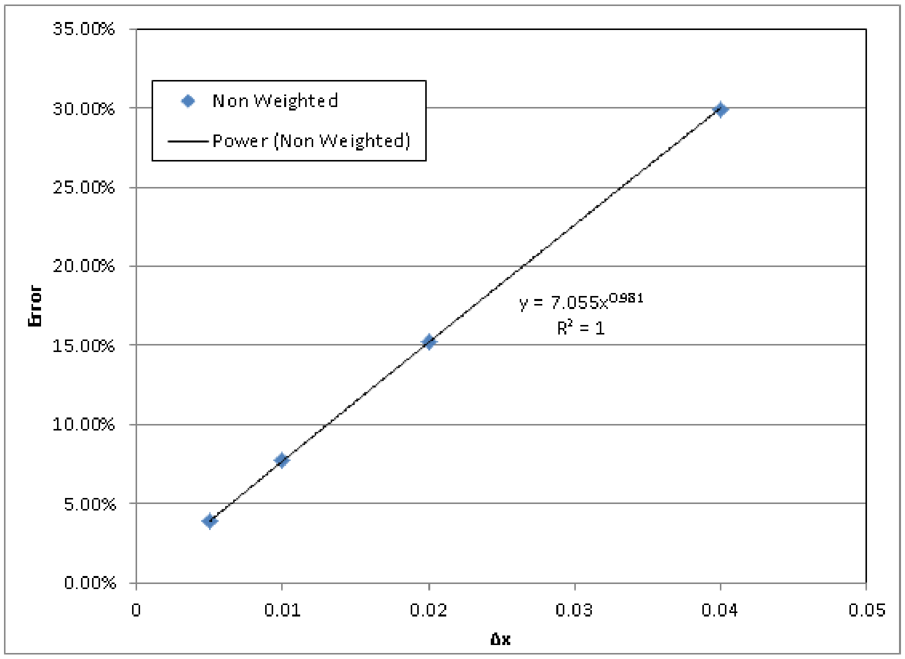


Figure 155: Coggeshall 19 direct method, two dimensional, density, unweighted L1 convergence.

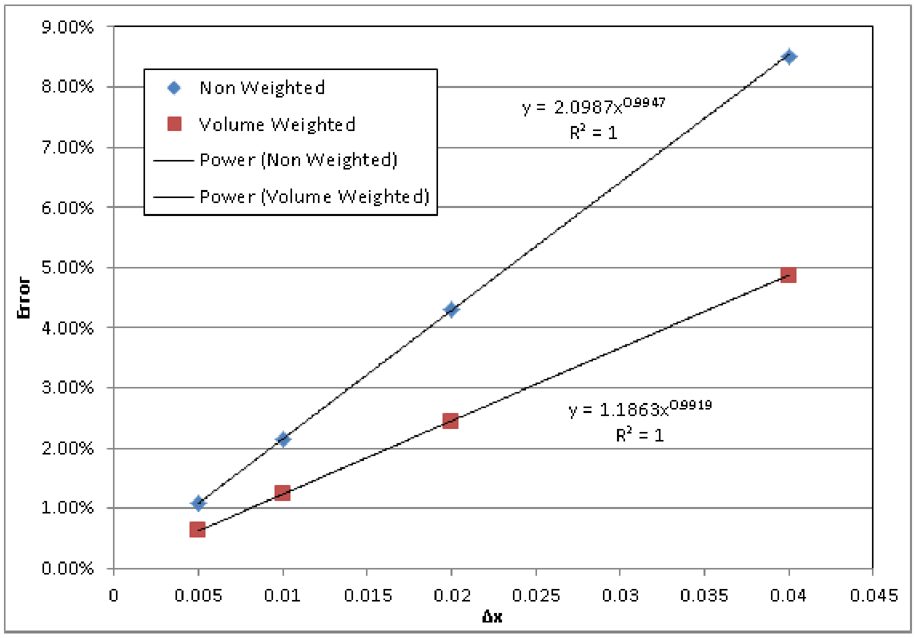


Figure 156: Coggeshall 19 piston method, two dimensional, density L1 convergence. Unweighted and volume-weighted norms are both plotted.

Table 6: Coggeshall 19 two dimensional direct method convergence summary.

Direct Method			
	Weighting	Convergence Constant	Convergence Rate
Density	None	7.0550	0.9810
Pressure	None	1.5656	0.9944
Temperature	None	0.2723	0.9349
Velocity	None	0.4300	0.9589

Table 7: Coggeshall 19 two dimensional piston method convergence summary.

Piston Method			
	Weighting	Convergence Constant	Convergence Rate
Density	None	2.0987	0.9947
	Volume	1.1863	0.9919
Pressure	None	0.1717	1.0006
	Volume	0.1127	1.0003
Temperature	None	0.1444	0.9955
	Volume	0.0794	0.9916
Velocity	None	0.3781	0.9733

Coggeshall 20

Coggeshall's problem 20 (Cog20) modelled a strong diverging shock with no conduction, and is a Lie Group transformation of Cog19 [12, 13, 53]. It was found that the problem did not satisfy the Rankine-Hugoniot jump conditions with the given shock location, so a different shock location was derived from the Rankine-Hugoniot jumps condition for a strong shock (see Appendix B of this paper for details). The new Cog20 equations (with free parameter values plugged in) are given by:

$$Post-Shock\,Equations:\ \rho(r,t)=\frac{64}{(1+t)^3},\ u(r,t)=\frac{(r)}{(1+t)},\ T(r,t)=\frac{1}{24(1+t)^2} \tag{179}$$

$$Pre-Shock\,Equations:\ \rho(r,t)=\frac{(r+t)^2}{(1+t)^3r^2},\ u(r,t)=\frac{(r-1)}{(1+t)},\ T(r,t)=0 \tag{180}$$

$$Shock\,Location:\ R(t)=\frac{t}{3} \tag{181}$$

$$Piston\,Location:\ P(t)=\frac{(1+t)}{4} \tag{182}$$

The shock given had free parameters a, k (geometry facor), ρ_0, and u_0, as well as constants γ and Γ, which are the adiabatic constant and the gas constant, respectively. We used a = -1, k = 2 (one dimensional spherical geometry), ρ_0 = 1, and u_0 = -1. With these restrictions, the shock always fulfilled the physical limitations of positive density and energy. We modelled a gas with $\gamma = \frac{5}{3}$ and $\Gamma = 8$. These parameters create a diverging sphere of high velocity, temperature, pressure, and density moving into a low density area with zero energy. The sphere is symmetric about the origin.

One Dimensional Spherical Results

For the direct computation, the given Cog20 regions were entered as the fields in the FLAG input deck. The code then calculated the progress of the shock through such an area, as can be seen in Figures 157 - 159 below.

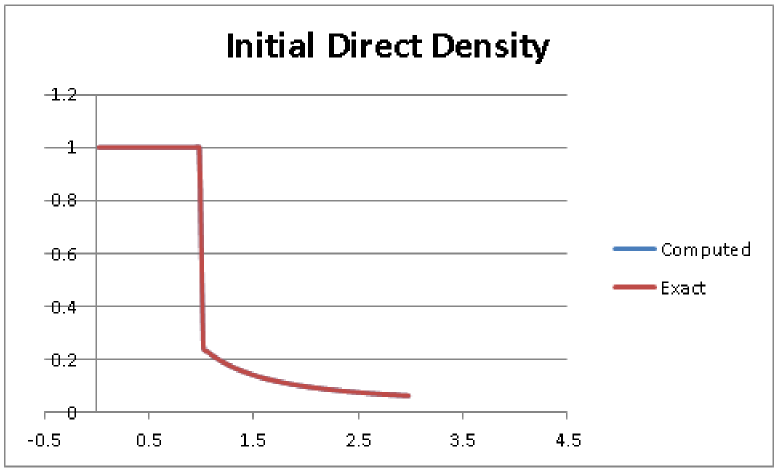


Figure 157: Coggeshall 20 direct method, initial density profile.

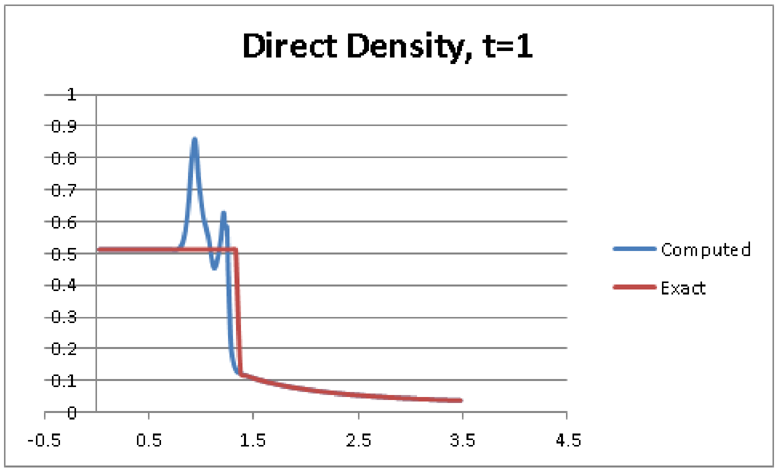


Figure 158: Coggeshall 20 direct method, density profile after one time unit.

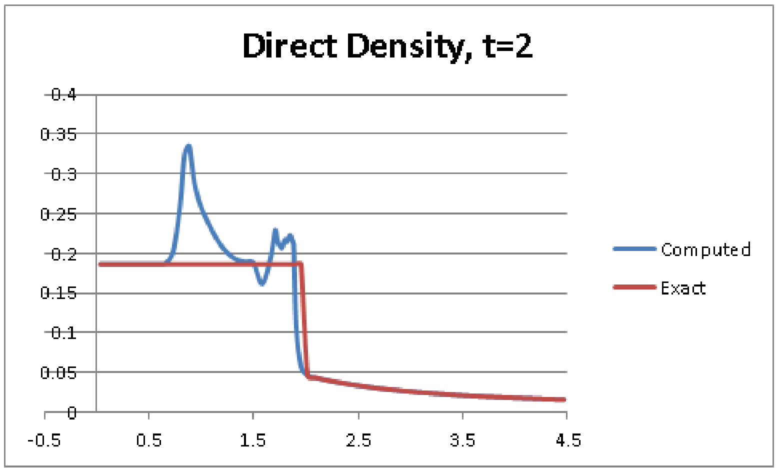


Figure 159: Coggeshall 20 direct method, density profile after two time units.

This was for a mesh with grid spacing 0.04. We repeated the simulation with grid spacing 0.02, 0.01, and 0.005, and theses results can be seen in our resolution plot, Figure 160. Note that we have focussed on the area around the shock, and that there are some oscillations in the simulation results as they approach the shock. Also, the shock is smeared out instead of being a precise discontinuity. These start up errors are common in shock modelling, as discussed earlier [36]. Turning our attention to the shock location, we observe that the simulated location approaches the actual with increasing mesh refinement. However, the shock is still smeared out over approximately the same number of cells, these cells are just growing smaller.

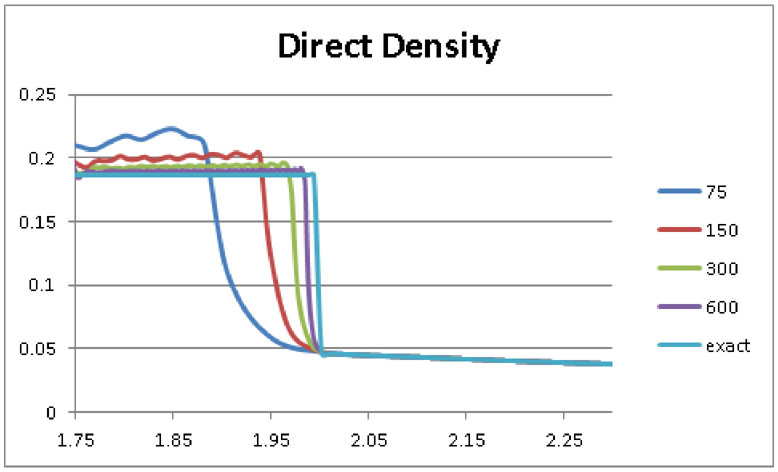


Figure 160: Coggeshall 20 direct method, density viewgraph for three time units.

Figure 161 shows the convergence study for density using the direct method by showing the L1 error norms against the mesh resolution. This figure gives quantitative data on the error of the code. First, we get the correlation constant. This is the constant multiplying x, and it roughly relates to the error of the simulation. Note that the unweighted error norms are less accurate than the mass and volume weighted methods. Second, we have the convergence rate, which is the order of the exponential plot. It relates to the order of accuracy of the code, which is first order in cases involving shocks. In fact, of all numerical methods for simulating shocks currently available, first order accuracy is the best that can be expected.

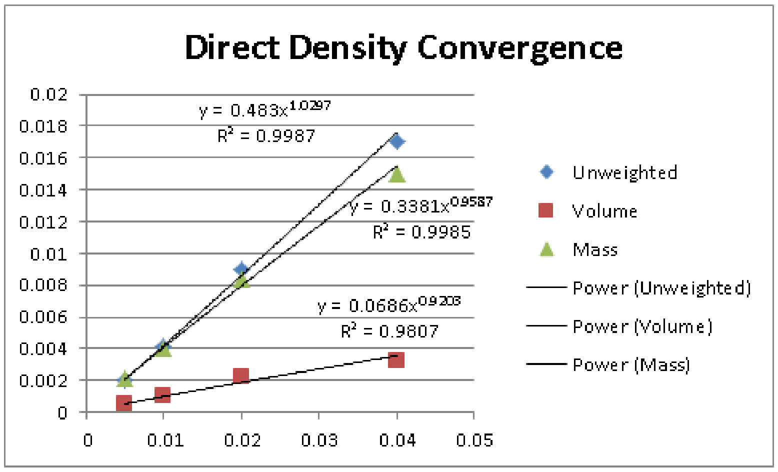


Figure 161: Coggeshall 20 direct method, density convergence for different weighting methods.

With the piston method, we first needed to find the motion of the piston that would simulate the desired shock. We started with an initial time where there was no shock and entered the unshocked region's conditions for our starting field. At this initial time, the piston and shock were lined up together at the boundary, and there was no shocked region. However, a point lined up with piston and shock would be the first particle in the shocked region once the simulation began. We knew from Coggeshall's equations how this point would move, so we knew how the leftmost boundary of the shocked region was going to move as well. This boundary is the "piston".

We found this particle's position

$$r(t) = c(1-at) \tag{183}$$

by solving the differential equation

$$u[r(t),t] = \frac{dr(t)}{dt} = -\frac{ar(t)}{1-at} \tag{184}$$

The constant c is found by equating the particle position and the shock position at some initial time, t_i.

$$c(1-at_i) = -\frac{1}{2}(\gamma-1)u_0t_i \tag{185}$$

$$c = \frac{(1-\gamma)u_0t_i}{2(1-at_i)} \tag{186}$$

With this constant, c, we now have the complete equation for the motion of the piston:

$$P(t) = \frac{(1-\gamma)u_0t_i(1-at)}{2(1-at_i)} \tag{187}$$

As with the direct method, we ran the simulation with grid spacing 0.04, 0.02, 0.01, and 0.005. In Figure 162 below, it is obvious that, as before, the simulation's accuracy improves with resolution. This is confirmed in the convergence plot, 163. The plot gives convergence rates of 0.9353, 0.9323, and 0.8969, which indicate first order accuracy, as expected. The correlation constants are smaller than with the direct method, giving 0.1332, 0.0968, and 0.0161 instead of 0.483, 0.3381, and 0.0686 for unweighted, volume-weighted, and mass-weighted calculations respectively. These smaller numbers for the piston method indicate an overall improvement in accuracy.

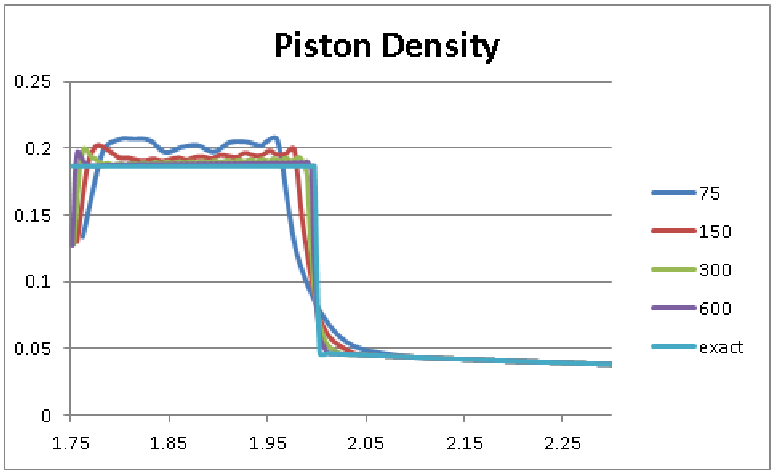


Figure 162: Coggeshall 20 piston method, density viewgraph at three time units.

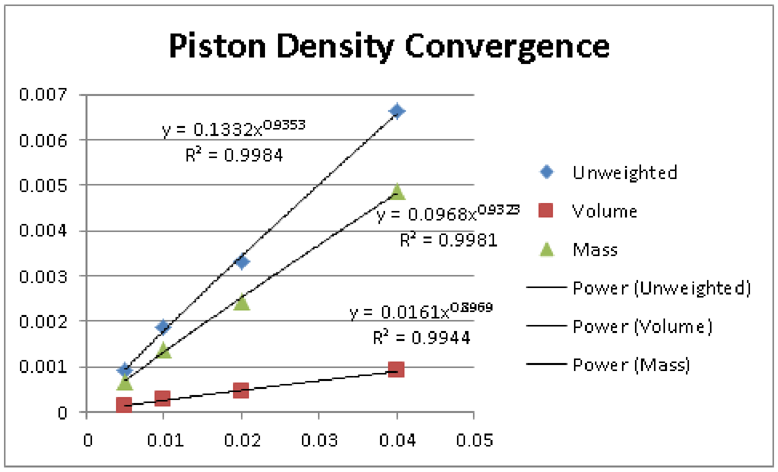
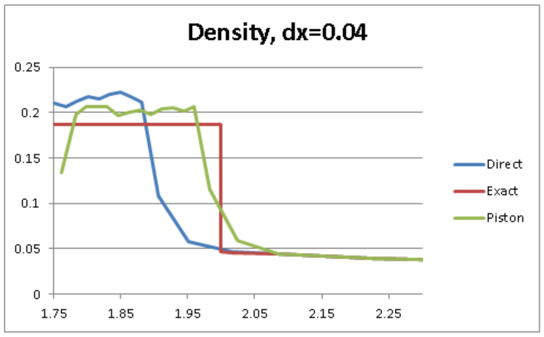


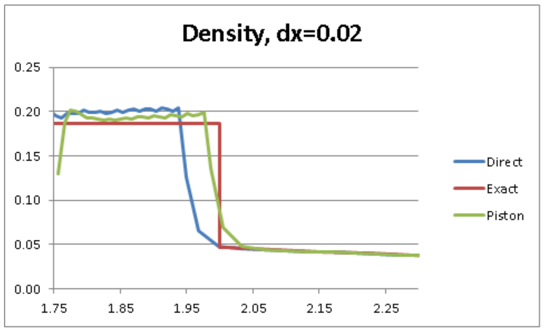
Figure 163: Coggeshall 20 piston method, density convergence.

Additionally, while there are still oscillations in the shocked region, they have been reduced. The smearing of the shock wave has lessened as well. However, we do have an issue with the left most boundary, a condition commonly referred to as wall-heating, which occurs with the sudden starting motion of the piston [47]. This sudden motion generates additional heat, which shows up as a dip in the density graph.

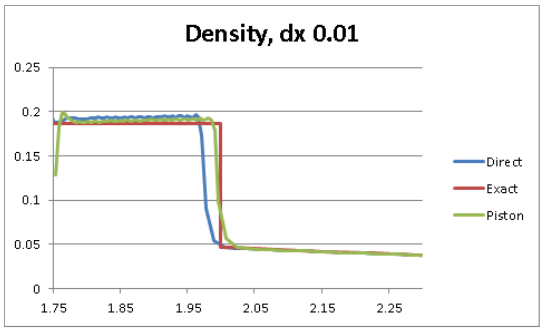
The Figure 164 below shows the direct and piston methods directly compared with the exact solutions for the resolutions simulated. At each resolution, the piston method is closer to the exact solution than the direct method. These plots confirm the earlier indications from the convergence constants, revealing that the piston method gives more accurate results.



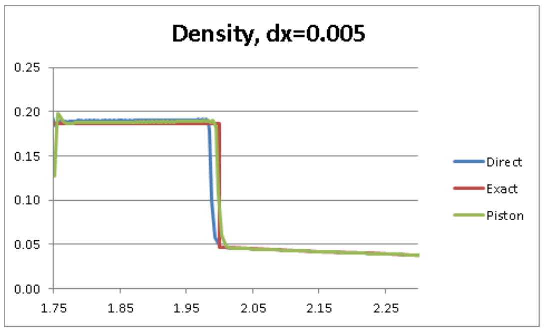
(a) dx=0.04



(b) dx=0.02



(c) dx=0.01



(d) dx=0.005

Figure 164: Coggeshall 20 direct and piston method direct density comparison.

Conclusion

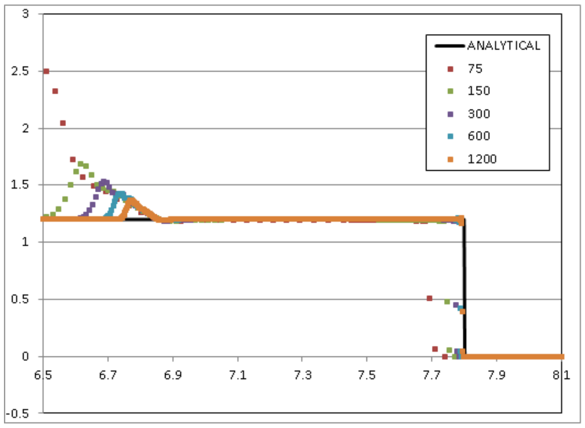
The convergence studies for both problems investigated revealed first order convergence for all variables. Since the convergence rates are expected to be first order in shock simulations, this provides some quantitative certainty in the accuracy of the numerical methods implemented. However, both problems were run using the same time step; the convergence studies did not take temporal convergence into consideration. Even though these problems do match up with the expected answer, it could be a fluke. To make sure the results are genuine, the problems should be run with even more mesh resolutions and on different test problems, as well as with different constant time steps. Verification is a never ending process. Each result can only increase the certainty of the numerics but can never fully prove the accuracy.

It is also recommended that a truly two dimensional analytical solution be tested as well. Even though the two dimensional test case presented did agree with the expected results, it was a quasi-one dimensional flow. The fluid only moved in x-direction and there was not any variation in the y-direction (except for computational noise). This test only checked to make sure that the x-components of the multidimensional numeric algorithms was verified.

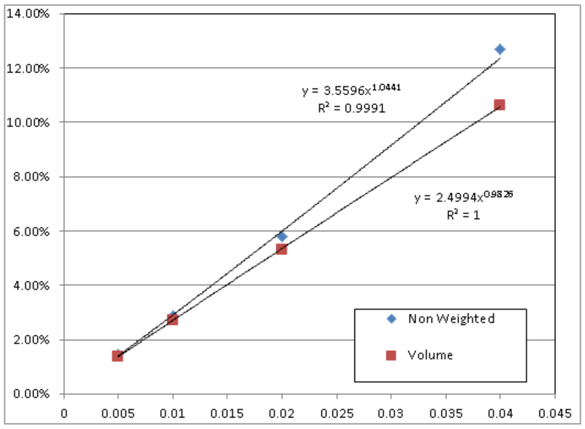
The convergence studies performed also showed the piston method to be more accurate than the direct method. Again, further test cases should be done, such as a case that does not assume strong shocks, since the two cases looked at in this study were very similar to each other. However, just from the test cases studied here, it is recommended that the piston problem be used whenever possible, since it is more accurate and more simple to set up the test domain. Unfortunately, the piston problem cannot be used in all cases, such as stagnation shocks or on Eularian codes. Therefore, the direct method is, currently, a more robust way of initialization.

Appendix A

Further Cog19 Direct Method Results

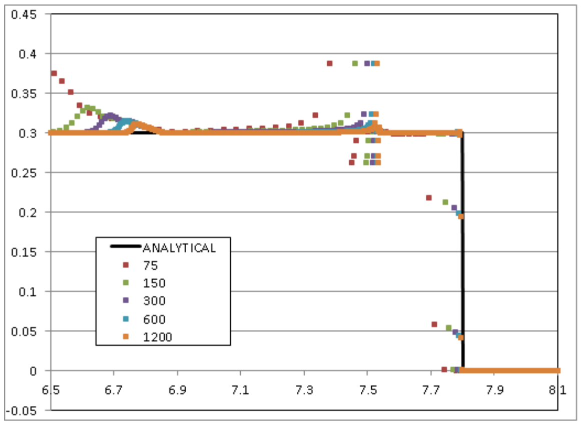


(a) Viewgraph norm.

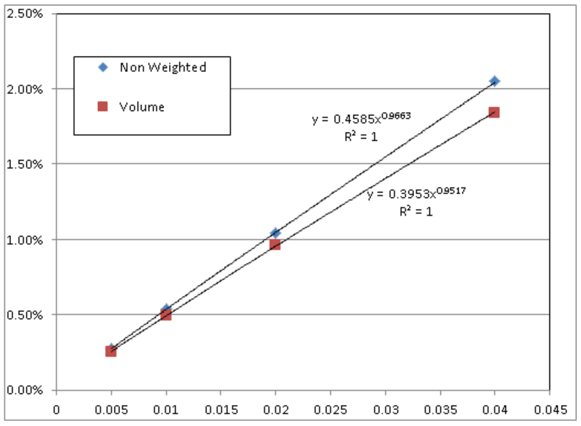


(b) Convergence plot.

Figure 165: Direct method: Pressure.

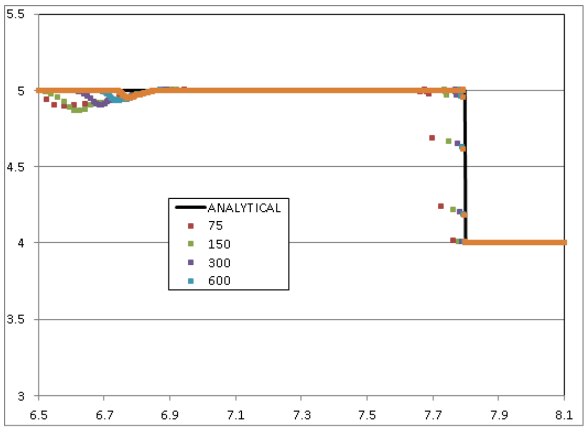


(a) Viewgraph norm.

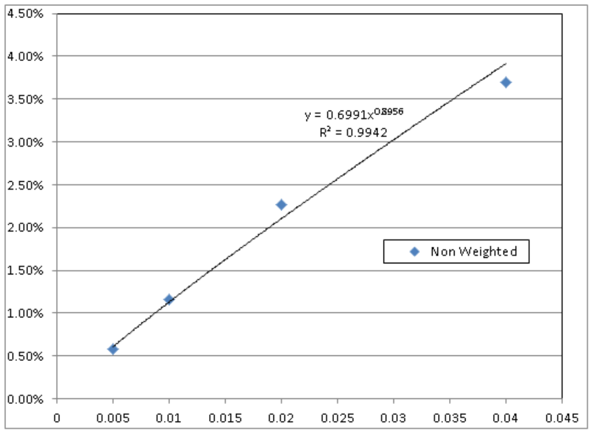


(b) Convergence plot.

Figure 166: Direct method: Temperature.



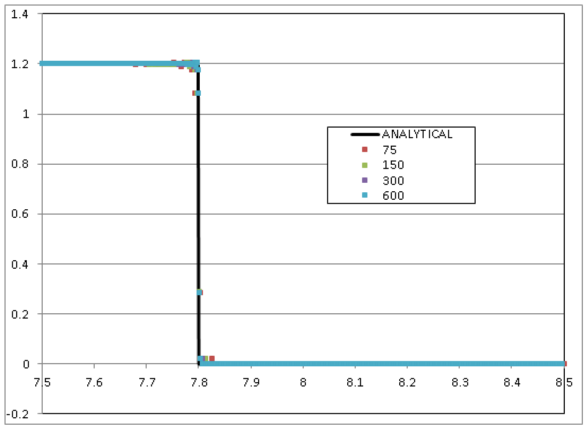
(a) Viewgraph norm.



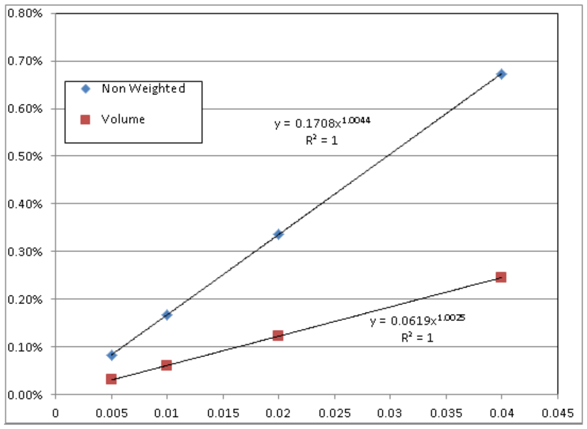
(b) Convergence plot.

Figure 167: Direct method: Velocity.

Futher Cog19 Piston Method Results

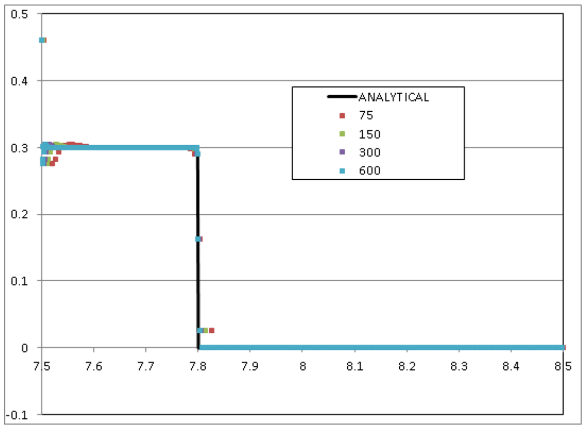


(a) Viewgraph norm.

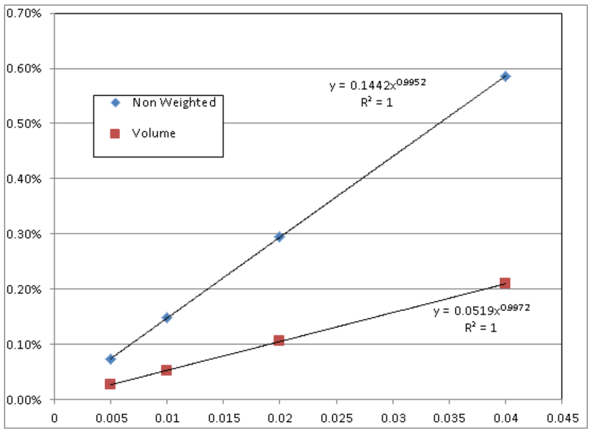


(b) Convergence plot.

Figure 168: Piston method: Pressure.

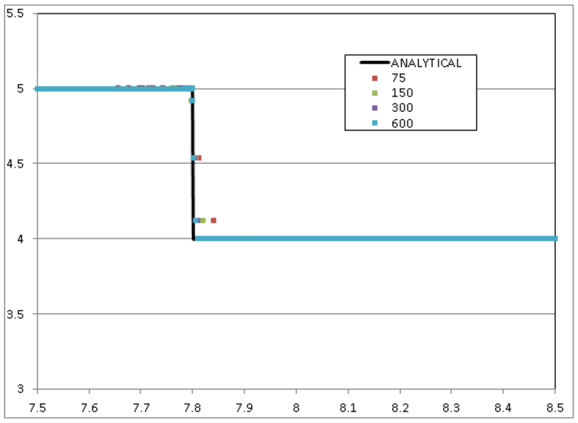


(a) Viewgraph norm.

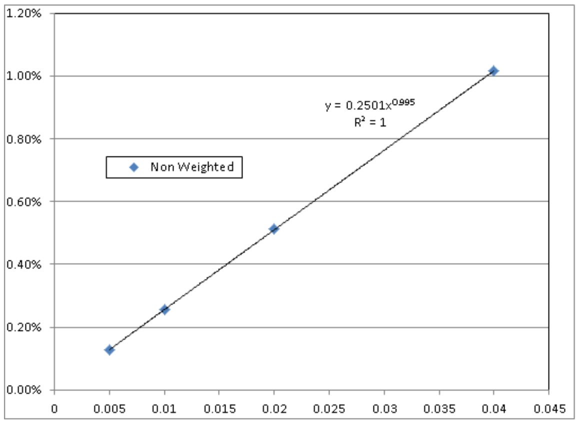


(b) Convergence plot.

Figure 169: Piston method: Temperature.



(a) Viewgraph norm.



(b) Convergence plot.

Figure 170: Piston method: Velocity.

Appendix B

Coggeshall gave

$$R(t) = \frac{u_0 t(\gamma - 1)(1 - 2at)}{4a(1 - at)} \tag{188}$$

as the shock location for Problem 20. However, it was discovered that with this shock location the problem did not fulfill the Rankine-Hugoniot jump condidtions for a strong shock. As a result, a different shock location for Cog20 was derived from the jump conditions.

The Rankine-Hugoniot jump conditions are:

$$\rho_1 u_1 = \rho_2 u_2 \tag{189}$$

$$p_1 + \rho_1 {u_1}^2 = p_2 + \rho_2 {u_2}^2 \tag{190}$$

$$w_1 + \frac{1}{2}{u_1}^2 = w_2 + \frac{1}{2}{u_2}^2 \tag{191}$$

where w is a heat function related to pressure and specific internal energy, and regions 1 and 2 are the unshocked and shocked regions respectively. Additionally, for a strong shock, we must have:

$$\frac{\rho_1}{\rho_2} = \frac{(\gamma-1)}{(\gamma+1)} \tag{192}$$

$$\frac{p_2}{p_1} = \infty \tag{193}$$

$$\frac{T_2}{T_1} = \infty \tag{194}$$

From these conditions, with a velocity transform we arrive at the differential equation

$$(u_1 - D)(\gamma-1) = (u_2 - D)(\gamma+1) \tag{195}$$

$$\frac{dR[t]}{dt} = D = \frac{\frac{(\gamma+1)}{(\gamma-1)}u_2 - u_1}{\frac{(\gamma+1)}{(\gamma-1)} - 1} \tag{196}$$

By plugging in the field velocities u_1 and u_2 from Cog20, with $r[t] = R[t]$, since we are examining the solution at the shock location, we obtain

$$\frac{dR[t]}{dt} = \frac{\frac{(-aR[t])(\gamma+1)}{(1-at)(\gamma-1)} - \frac{u_0-aR[t]}{(1-at)}}{\frac{(\gamma+1)}{(\gamma-1)} - 1} = \frac{2aR[t] + (\gamma-1)u_0}{2(at-1)} \tag{197}$$

This ODE was found to be solved by:

$$R[t] = -\frac{(\gamma-1)u_0}{2a} + c(at-1) \tag{198}$$

If we say that $R[0] = 0$, then

$$R[t] = -\frac{(\gamma-1)u_0 t}{2} \tag{199}$$

Using the chosen free parameter value of $u_0 = -1$ and constant $\gamma = \frac{5}{3}$, we obtain

$$R[t] = \frac{t}{3} \tag{200}$$

the shock location used in our simulations.

Improving Plasticity Modeling in Hydrocodes with Hypoelastic Frameworks

By Nathan Walter[1], Paul Friedrichsen[2], and Scott Runnels[3] (mentor)

[1] University of Illinois Champaign-Urbana Department of Nuclear Engineering, [2] St. Olaf College Department of Physics, [3] Los Alamos National Laboratory

Abstract

The current method of numerically simulating high strain rate plastic flow, Wilkins method or radial return, can experience inaccuracies and be non-convergent in some cases. The method is a discrete algorithm for calculating the stress tensor, which is often coupled with an ad hoc method of advection, an inaccurate method of handling finite body rotations, and assumed independent of a changing yield surface. In this report, an alternative solution to these issues is implemented. The stress tensor is calculated with an exact equation in strain space. Consideration of finite body rotations is reexamined. The changing values of strain and flow stress are coupled into a single solution. These alternative methods are used in a production code, FLAG, to compare results to the radial return method. From the results, the solutions to numerical solid dynamic simulations are found to be stable, accurate, convergent, and efficient.

Introduction

A hypoelastic framework for a computational solid mechanics code is one in which the stress tensor is expressed as a function of strain. That framework stands in contrast to a hyperelastic framework in which stress is expressed as a gradient of energy, specifically, in terms of partial derivatives of the material's energy with respect to the deformation gradient tensor, which is often denoted by F_{ij}. The hyperelastic framework has the advantage of offering a superior way of handling large-scale deformations. Also, for ALE calculations, the hyperelastic framework offers the advantage of being built upon an entity that, apparently, can be advected with confidence. Specifically, the hyperelastic framework is built on the deformation gradient tensor, F_{ij}. There appears to be a conservation law for the inverse of F_{ij}. Therefore, in ALE, when the mesh relaxes and variables are advected, the inverse of F_{ij} is advected accurately, used to reconstruct F_{ij}, and then from that the stress tensor is computed. In contemplating the application of the hyperelastic framework to plasticity, additional complexities arise in the aforementioned gradient calculation. It is necessary that the gradient calculation deal with all of the same aspects of plasticity that traditional approaches, i.e., those built on the hypoelastic framework, have addressed through algorithms such as radial return. Those complexities have been worked out to some extent and are being used in some state-of-the-art computational solid mechanics solvers.

However, there are important solid mechanics solvers that are built on the hypoelastic approach, and have evolved into hypoplasticity codes using algorithms such as radial return. CTH is one well-known example [28]. The difficulties encountered by those codes have to deal with large-scale deformation. In the 1980s, substantial progress was made when investigators made use of the polar decomposition theorem to develop ways to approximately transform between the material and laboratory reference frame, thereby in principle, eliminating the effects of large-scale deformation. In particular, the Green-Nahgdi method of computing the rigid body rotation tensor in the polar decomposition of the deformation gradient tensor greatly improved accuracy in some situations [20]. However, the problem of advection for ALE calculations still remained. When mesh relaxation occurs and a remap of variables is required, it is not altogether clear how to reconstruct a full stress tensor from material once in difference cells but now combined into one cell. Unlike the inverse of F_{ij}, there is no conservation law for a stress tensor and one is left with the need to develop some type of ad hoc method. In some codes, the stress tensor is computed using a volume weighted average of the material pieces. In other codes, mass weighting is used.

In the work described here, a different way to address the advection problem in hypoplastic codes is implemented and tested. However, that aspect is not the only motivator for the underlying algorithms that are studied. Specifically, there are other issues, even in one-dimensional problems with strain-rate hardening materials, with the traditional hypoplastic approach, using radial return. As shown by Margolin [41], there are issues with time steps and stability when the material when the yield surface changes with strain and strain rate. Also, as demonstrated by Runnels, traditional radial return, when run explicitly without nonlinear iteration between equivalent plastic strain and yield stress, results in spatial non-convergence. Others have found that nonlinear iteration on the yield surface is important, presumably helping with non-convergence issues that arise when one does not iterate. However, the nonlinear iteration must occur between the radial return algorithm and a set of equations for the material hardening. The algorithm studied in this report represents the radial return algorithm as an ordinary differential equation, the analytical solution of which exists and is known; this was the work of Margolin and Flower, as described in [40] and [41]. Thus the pre-existing situation of a radial return algorithm iterating with an analytical hardening rule may be replaced with a more structured approach of two analytical nonlinear equations solved simultaneously. The improved structure therefore admits more advanced and effective nonlinear iteration algorithms. It also allows for the material modeling equations, which includes the exact return ODE, the hardening equations, and the hydro conservation equations to exist together, simultaneously, in the same mathematical framework. Couching them together, in that way, admits new possibilities for more effective, convergent, and accurate overall numerical methods.

The outline of this chapter is as follows. In the next section, the background and necessary basic concepts of the material is introduced. Then, the current implementation of the discrete radial return method is described. After that, a new method for modeling plasticity is introduced with a constant flow stress. In that section, the governing equations are derived, considerations needed for implemented are presented, and results with the method are shown. Next, a new method of computing the advected value of the stress tensor is presented. In the section following that, a new method for modeling plasticity is introduced with rate hardening included. In that section, the governing equations are derived, considerations needed for implemented are presented, and results with the method are shown. Then, the exact implementation of the PTW model is studied. Finally, the addition of finite body rotations is presented.

Background

Starting Concepts

To begin, the stress tensor in three dimensions can be split into two sub tensors.

$$\begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix} = - \begin{bmatrix} p & & \\ & p & \\ & & p \end{bmatrix} + \begin{bmatrix} s_{11} & s_{12} & s_{13} \\ s_{21} & s_{22} & s_{23} \\ s_{31} & s_{32} & s_{33} \end{bmatrix} \tag{201}$$

where p is the average of the normal stresses and s_{ij} is the deviation from the average. The average bulk pressure is defined as the average of the normal direction stress or one third the trace.

$$p = -\frac{1}{3}\mathrm{tr}(\sigma_{ij}) \tag{202}$$

There are two regimes of deformation, elastic and plastic. When a material is being elastically loaded, the material will return to its normal shape when the force is unloaded. In this regime, the deviatoric stress obeys the following rule

$$s_{ij} = 2Ge_{ij} \tag{203}$$

When a material is in the plastic regime, the strain accumulated by the loaded stress is only partially reversible. In this regime, the relation between deviatoric stress and deviatoric strain rate is more complicated. Here we introduce the concept of a yield surface. The concept is easier to understand in 1-D. The plot in Figure 171 is that of a simple elastic perfectly plastic stress-strain curve in 1-D

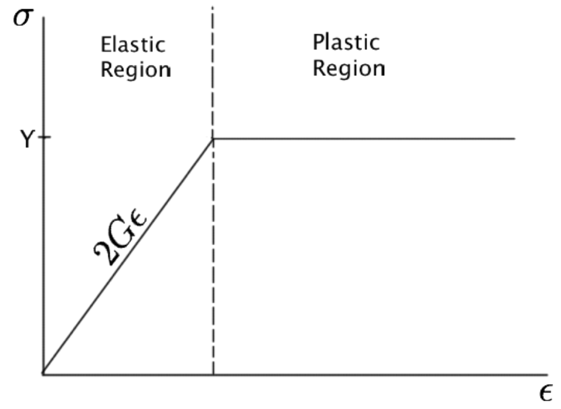


Figure 171: This is a stress strain curve in 1-D. This figure shows the elastic and plastic regime. σ is stress and ε is strain

In this plot, the stress increases linearly with strain until the stress is equal to the yield stress, Y. The stress of the system can never exceed the yield surface, defined by what is often referred to as the yield stress or sometimes the flow stress. Once the stress is equal to Y, the system begins to flow or deform, which relaxes the system causing a new relation between strain and stress. Meaning, the material composition begins to change, allowing more plastic strain to accumulate, but the stress does not exceed Y. Obviously, this is a simple case. Y can vary with plastic strain, time, and other variables. In higher dimensions, instead of a flow stress in each of the nine directions, there is still only one Y. Thus, the concept of a stress invariant must be used. J_2 is one of the most popular stress invariants to use. It is defined as follows

$$J_2 = \frac{1}{6}\left[(s_{11}-s_{22})^2+(s_{22}-s_{33})^2+(s_{33}-s_{11})^2\right]+s_{12}^2+s_{23}^2+s_{31}^2 \tag{204}$$

Using this invariant, the criteria of material flow can be determined by the following relation. A material must obey

$$J_2 \leq \frac{Y^2}{3} \tag{205}$$

These are the foundational concepts for the next sections.

Discrete Radial Return

Many if not most Lagrangian hydrocodes that accommodate plasticity implement the concepts described above through a method called radial return [?]. Here, we prepend that term with the word “discrete” for reasons that will become clear below. During a given time step in a hydro code, a deviatoric strain rate is calculated from the change in each cell’s shape. Thus, the strain rate for each cell is a value provided by the hydro algorithm that is calling the plasticity model. Furthermore, that rate is often considered to be a constant over the time step being considered. A two-step process is used for calculating the deviatoric stress. The first step is to compute a trial value via an elastic extrapolation, as follows:

$$s_{ij}^* = s_{ij}^0 + \sum_{n=1}^{N} 2G\dot{e}_{ij}^n \Delta t^n = s_{ij}^{N-1} + 2G\dot{e}_{ij}^N \Delta t^N \tag{206}$$

Here, N represents the current step; N-1 is the previous time step. 0 represents an initial state, and 1 is the state after one time step. s_{ij}^* represents that this is only a trial deviatoric stress value. From the trial values, a trial stress invariant is calculated. If the material has not yielded yet, then the trial values are accept as the true values. If the material has yielded, the stress values are scaled such that, the invariant is equal to the flow stress

$$\begin{cases} s_{ij}^N = s_{ij}^* & J_2 \leq \frac{Y^2}{3} \\ s_{ij}^N = \alpha s_{ij}^* & J_2 > \frac{Y^2}{3} \end{cases} \tag{207}$$

where

$$\alpha = \frac{Y}{\sqrt{3J_2}} \tag{208}$$

This process can be visualized in Figure 172.

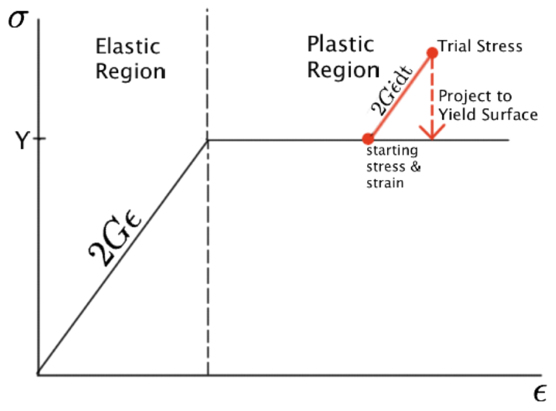


Figure 172: Visualization of Discrete Radial Return Method

This Figure shows how the stress is elastically incremented, and then projected back to the yield surface.

Elastic Perfectly Plastic Case

Equations with $\dot{Y}$=0

The equations for discrete radial return in the previous section when combined and the limit of Δt is then taken to very small values, an exact ODE for the radial return method is produced [39]. This process has been displayed in L.G. Margolin's paper "A strain space framework for numerical hyperplasticity," where he then showed the result to be the differential equation

$$\boxed{\begin{cases} \frac{ds_{jk}}{dt} = 2G\left(\frac{de_{jk}}{dt} - \frac{3W}{2Y^2}s_{jk}\right) & J_2 > \frac{Y^2}{3} \\ \frac{ds_{jk}}{dt} = 2G\frac{de_{jk}}{dt} & J_2 \leq \frac{Y^2}{3} \end{cases}} \tag{209}$$

where W is a scalar value of

$$W = s_{jk}\frac{de_{jk}}{dt} \tag{210}$$

These two differential equations can be exactly integrated over a time step. It is because of this ability to exactly integrate the governing equations that the process described in section Discrete Radial Return is coined the discrete radial return. The final equations in this section will be deemed the Exact Return because their origin comes exact integration. The process of exactly integrating the differential equations is as follows:

The elastic case can be clearly integrated and be shown to produce the same results presented in the previous section. The plastic regime differential equation is as follows

$$\frac{ds_{jk}}{dt} = 2G\left(\frac{de_{jk}}{dt} - \frac{3W}{2Y^2}s_{jk}\right) \tag{211}$$

If this equation is multiplied by strain rate, the equation becomes

$$\frac{dW}{dt} = 2G\left(I^2 - \frac{3W^2}{2Y^2}\right) \tag{212}$$

which can be solved to be

$$W(t) = \frac{Y^2}{3G}\frac{d}{dt}[\ln F] \tag{213}$$

where

$$F(t) = A_o \exp(\alpha t) + \exp(-\alpha t) \tag{214}$$

By using this definition of W in Equation 211, the differential equation for stress in the plastic regime can be solved as

$$s_{jk}(t) = s_{jk}(0)\frac{F(0)}{F(t)} + \frac{2G}{F(t)}\frac{de_{jk}}{dt}\int_0^t F(t')dt'$$

which is solved to become

$$\boxed{s_{jk}^{n+1} = s_{jk}^{n}\left[\frac{\chi_n(A_n+1)}{A_n\chi_n^2+1}\right] + \frac{\sqrt{2}Y}{\sqrt{3}I^n}\frac{de_{jk}^n}{dt}\left[\frac{A_n\chi_n^2 - 1 - \chi_n(A_n-1)}{A_n\chi_n^2+1}\right]} \tag{215}$$

where

$$\chi_n = \exp(\alpha_n\Delta t); \quad \alpha_n = \sqrt{6}I_n\frac{G}{Y}; \quad A_n = \frac{\sqrt{2}I_nY + \sqrt{3}W_n}{\sqrt{2}I_nY - \sqrt{3}W_n} \tag{216}$$

$$I_n = \sqrt{\frac{de_{jk}^n}{dt}\frac{de_{jk}^n}{dt}}; \quad W_n = s_{jk}\frac{de_{jk}^n}{dt} \tag{217}$$

In order to represent this equation in more simplified terms, the following alternate equation is presented

$$\boxed{s_{jk}^{N+1} = s_{jk}^1 + \left[\sum_{n=1}^{N} 2G\dot{e}_{jk}^n\Delta t^n - \sum_{n=1}^{N}\mathscr{I}^n\right]} \tag{218}$$

By comparing the Equations 218 and 215 to one another, the definition of $\mathscr{I}^n$ can be deduced as

$$\boxed{\mathscr{I}^n = s_{jk}\left(1 - \frac{\chi_n(A_n+1)}{A_n\chi_n^2+1}\right) + 2G\dot{e}_{jk}^n\Delta t + \frac{\sqrt{2}Y}{\sqrt{3}I_n}\frac{de_{jk}^n}{dt}\left[\frac{A_n\chi_n^2 - 1 - \chi(A_n-1)}{A_n\chi_n^2+1}\right]} \tag{219}$$

With these new definitions, the total strain can be written as

$$e_{jk}^T = \frac{1}{2G}\left(s_{jk}^1 + 2G\sum_{n=1}^{N}\dot{e}_{jk}^n\Delta t^n\right)$$

and the total plastic strain can be written as

$$e_{jk}^P = \frac{1}{2G}\sum_{n=1}^{N}\mathscr{I}^n$$

Thus making the final solution for stress as follows

$$\boxed{s_{jk}^{N+1} = 2G\left(e_{jk}^T - e_{jk}^P\right)} \tag{220}$$

This is the final governing equation for the method. Notice that the stress tensor is now an exact equation in strain-space. It is because of the exact integration of the differential equation and the final equation being an explicit exact equation that this method is referred to as the exact return.

Fractional Time

One of the major assumptions during the derivations of the exact return was that all time integrals were over a single Δt step, from 0 to t. This implies that the integrals were integrated under the assumption that the entire step was either yielded or not yielded. However, this is not a correct assumption. This can be seen in Figure 173, when the method was first implemented.

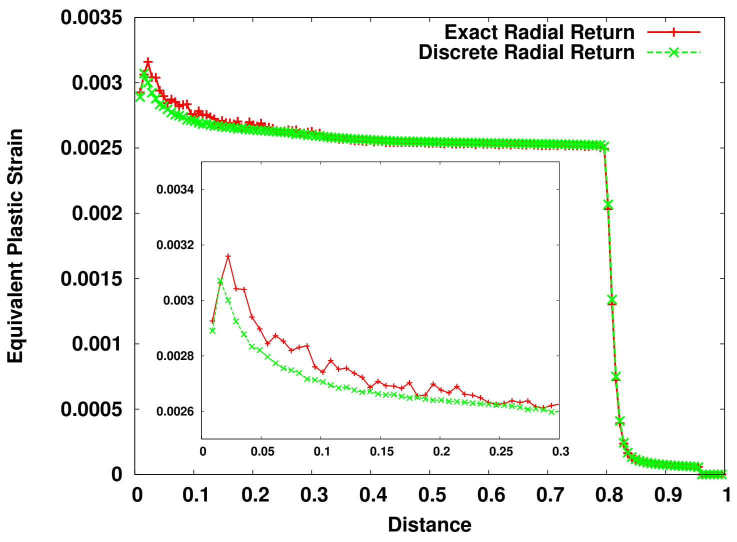


Figure 173: Equivalent plastic strain versus distance for a 1-D piston problem with elastic perfectly plastic material. Figure includes a zoom of the region with most error

The plot shows that some error accumulates when the material is assumed yielded during entire steps even if the material begin off the surface and ended on the surface. The source of error can be explained in the following Figure.

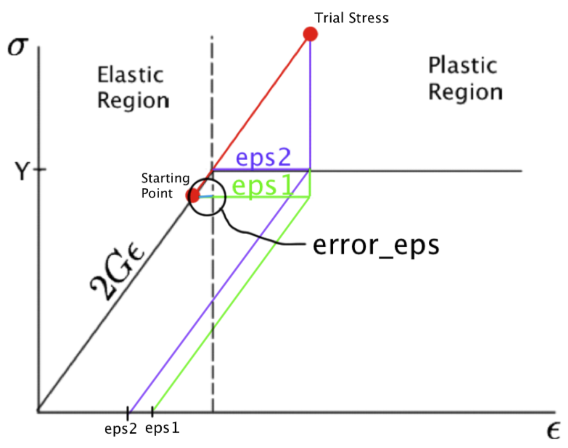


Figure 174: A graphical representation of the error produced when assuming yielded or not yielded during an entire step

Figure 174 illustrates why it is wrong to assume that the entire step is yielded if the trial stress is above the yield surface. The figure shows two methods of computing equivalent plastic strain. eps1 represents the equivalent plastic strain if the entire step time is assumed yielded, and eps2 represents the actual equivalent plastic strain. The figure shows that some amount of error is prevalent between eps1 and eps2. Thus, the error from the assumption must be accounted for. In order to reduce the error in the method, the integrals must be evaluated only during the fraction of the step on the yield surface. Therefore, all instances of time must be scaled by a constant f.

$$\Delta t \longrightarrow f\Delta t \tag{221}$$

where

$$f = \frac{\sqrt{3J_2^{trial}} - Y}{\sqrt{3J_2^{trial}} - \sqrt{3J_2^{old}}} \tag{222}$$

Because of the linear relation between time, strain, and stress, the fraction of time on the yield surface can be represented by the difference between stress invariant and the flow stress divided by the total change in the stress invariant, as shown by Equation 222. One can fully determine the value of f from the geometry of the figure above.

Results

With the method fully developed, it was implemented in the production code FLAG and in a simple 1-D c++ code. The method was then tested on a 1-D piston problem

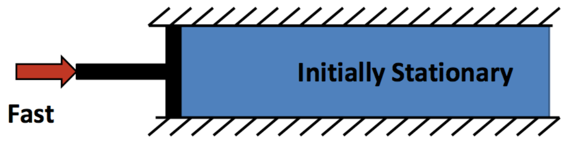


Figure 175: illustration of a 1-D piston problem

This figure shows the set up for the 1-D piston problem. The method was applied to this problem, and the equivalent plastic strain was calculated and plotted against the distance of the piston.

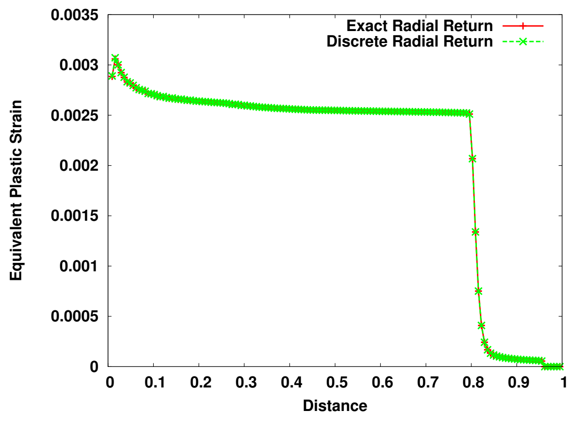


Figure 176: Equivalent plastic strain versus distance for a 1-D piston problem with elastic perfectly plastic material with consideration for fractional integral periods

This figure shows the equivalent plastic strain versus distance for a snapshot in time. The discrete radial return is the currently implemented method of radial return. From Figure , one can see that the Exact return and the discrete method produces point for point, essentially, the same line. This is to be expected as the discrete radial return produces correct solutions when run on a perfectly plastic material. Thus, this figure shows code to code verification of the Exact return, because both the c++ code and FLAG confirm the solution produced by the Exact return. With the Exact return verified, it was easy to pursue other topics with the method, such as advection, including rate hardening into the method, and 3-D capabilities.

Accommodating ALE

One of the possible benefits of the exact return formulation is that it is based on strain, which means that when ALE is executed, advection can be based on strain as opposed to stress as is typically done for hypoelastic codes that have been extended for plasticity. In this section we introduce a method for advecting the stress tensor through the strain tensor. The current method of handling ALE in FLAG is to advect stress, and that advectioin is a two step process. The current method uses volume weighting. This means the values of the are averaged based on the percent of the new volume they fill. First, the J_2 invariant is determined for each material in the zone. Then, the J_2 invariants and the stress components are volume weighted advected. Lastly, the J_2 invariant is computed from the volume averaged stress tensor. Then the new invariant is used to determine how much the volume-weighted stress tensor must be scaled in order to retain the volume weighted invariant.

$$J_2^{new} = VolA * J_2^A + VolB * J_2^B \tag{223a}$$

$$\sigma_{jk}^{trial} = VolA * \sigma_{jk}^A + VolB * \sigma_{jk}^B \tag{223b}$$

$$\sigma_{jk} = \sigma_{jk}^{trial} \sqrt{\frac{J_2^{new}}{J_2^{trial}}} \tag{223c}$$

Where the superscript $trial$ refers to a temporary value until the stress tensor is scaled based on the stress invariants. The major issue with this method of advection is the lack of conservation equation for stress. And there is no firm physical basis for volume weight advecting the stress invariant. While this method has sufficed as good enough for a long time, there is still a lack of sound science behind the method.

However, strain does have a conservation equation. Strain is conserved by volume. And strain can also be advected by volume with justification. Thus, both plastic and elastic strain can be advected in the exact return algorithm, and then an advected value of stress can be computed from the advected values of strain through Equation 220. Thus providing sound physics behind the advection of the stress tensor.

When this method, based on the advection of strain, was implemented, the method was again tested on a 1-D perfectly plastic piston problem. The results are shown in Figure 177, which shows the Exact Return with strain advected stress compared to the Discrete method with *ad hoc* advected stress, the following plot of the equivalent plastic strain was produced.

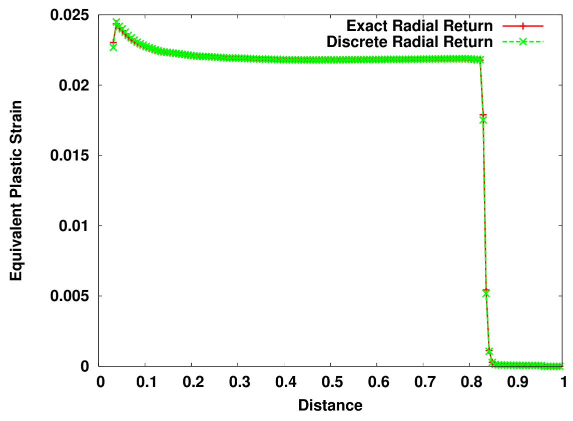


Figure 177: Equivalent plastic strain versus distance for a 1-D perfectly plastic piston problem. Exact Method is strain advected stress. Discrete method is ad hoc advected stress

This figure shows that the new Exact method with strain advected stress compares very well to the Discrete method with *ad hoc* advected stress. While the points are not necessarily in the same locations because of the different mesh relaxation actions, the lines are nearly identical. Again, since a perfectly plastic case is correct with the Discrete radial return, this figure verifies that the strain advected stress method is at least consistent with the stress advection method in

1-D. Similarly, the Exact return with strain advected stress can be compared to the discrete radial return in lagrange mode. The next plot shows this comparison.

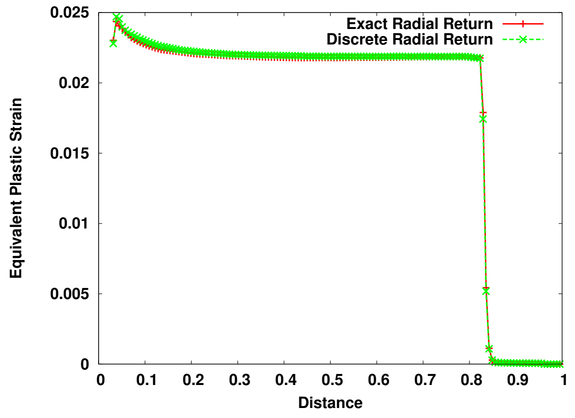


Figure 178: Equivalent plastic strain versus distance for a 1-D perfectly plastic piston problem. Exact Method is strain advected stress. Discrete method is in lagrange mode

Figure 178 shows that the new Exact return with strain advected stress compares very well to the lagrange mode of the Discrete radial return method. This further shows the consistency of the strain advected method with the stress advection method in 1-D.

Case Study for Rate-Hardening Materials

While a perfectly plastic material is a good test case, real materials are not perfectly plastic. Real materials have a flow stress that changes based on many variables. Real materials experience rate hardening, meaning the material becomes more resistant to flow with increasing plastic strain. Because of this, the method for the Exact Return was extended to include rate hardening materials. However, to include this, the governing differential equation for stress now changes. This section shows the derivation of the new governing equations for stress and strain, as well as introduces the need for nonlinear iterations.

Equations with $\dot{Y} \neq 0$

The previous section used the differential equation where the flow stress is constant. However, if the flow stress is not assumed constant, then the differential equation formed when the limit of Δt is taken is slightly different. This subsections derivation has been shown in [40]. From L.G. Margolin and E. Flower's paper "Numerical Simulation of Plasticity at High Strain Rate," the differential equation, as compared to Equation 209, becomes as follows when $\dot{Y}$ is not zero

$$\boxed{\frac{ds_{jk}}{dt} = 2G\frac{de_{jk}}{dt} - \frac{3GW}{Y^2}s_{jk} + \frac{1}{Y(t)}\frac{dY(t)}{dt}s_{jk}} \tag{224}$$

with W(t) remaining of the same form

$$W(t) = \frac{Y^2}{3G}\frac{d}{dt}\ln F$$

however F is now defined as

$$F(t) = A_o \exp(\alpha) + \exp(\alpha)$$

Notice the time variable, t, no longer in the exponential terms. This is due to the new definition of α

$$\alpha = \sqrt{6} \int_0^t \frac{GI}{Y(t')} dt'$$

If $\dot{Y}$ is assumed constant in time, which is a safe assumption as long as Δt for a step is very small. If this assumption holds then Y(t) can be expressed as a linear function

$$Y(t) = Y(0) + \dot{Y}(0)t$$

This allows the integral in alpha to be calculated as

$$\alpha = \frac{\sqrt{6}GI\Delta t}{Y(t) - Y(0)} \ln\left(\frac{Y(t)}{Y(0)}\right)$$

This definition of α holds in the limit as $\dot{Y}$ goes to zero. The equation above becomes the equation for α in the previous section for $\dot{Y}$ equals zero.

With these new conditions, the differential equation can now be solved to produce

$$s_{jk}(t) = s_{jk}(0)\frac{Y(t)}{Y(0)}\frac{F(0)}{F(t)} + \frac{2Y(t)}{F(t)}\frac{de_{jk}}{dt}\int_0^t \frac{F(t')G}{Y(t')}dt' = s_{jk}(0)\frac{Y(t)}{Y(0)}\frac{F(0)}{F(t)} + \frac{\sqrt{2}Y(t)}{\sqrt{3}IF(t)}\frac{de_{jk}}{dt}\int_0^{\alpha(t)} F(\alpha')d\alpha'$$

which can be integrated to become a comparison to Equation 215

$$\boxed{s_{jk}^{n+1} = s_{jk}^n \left[\frac{\chi_n (A_n + 1)}{A_n \chi_n^2 + 1}\frac{Y(t)}{Y(0)}\right] + \frac{\sqrt{2}Y(t)}{\sqrt{3}I^n}\frac{de_{jk}^n}{dt}\left[\frac{A_n\chi_n^2 - 1 - \chi_n (A_n - 1)}{A_n\chi_n^2 + 1}\right]} \tag{225}$$

where

$$\chi_n = \exp(\alpha_n \Delta t); \quad \alpha = \frac{\sqrt{6}GI\Delta t}{Y(t) - Y(0)} \ln\left(\frac{Y(t)}{Y(0)}\right); \quad A_n = \frac{\sqrt{2}I_n Y(0) + \sqrt{3}W_n}{\sqrt{2}I_n Y(0) - \sqrt{3}W_n}$$

$$I_n = \sqrt{\frac{de_{jk}^n}{dt}\frac{de_{jk}^n}{dt}}; \quad W_n = s_{jk}\frac{de_{jk}^n}{dt}$$

This concludes the derivation displayed in reference [40].

Extending the Derivation

In order to implement the rate hardening case in the same fashion as the perfectly plastic case, the derivation shown in [40] has to be extended. Thus the method used in the perfectly plastic case or the method used in [39] was used on the equations displayed above. This method is needed to get the equations from the previous subsection to be strain based.

Therefore, if the equation for stress is again represented as

$$\boxed{s_{jk}^{N+1} = s_{jk}^{1} + \left[\sum_{n=1}^{N} 2G\dot{e}_{jk}^{n}\Delta t^{n} - \sum_{n=1}^{N} \mathscr{I}^{n}\right]} \tag{226}$$

then by comparing Equations 225 and 226 yields the value for $\mathscr{I}^{n}$ to be

$$\boxed{\mathscr{I}^{n} = s_{jk}\left(1 - \frac{\chi_n\left(A_n+1\right)}{A_n\chi_n^2+1}\frac{Y(t)}{Y(0)}\right) + 2G\dot{e}_{jk}^{n}\Delta t + \frac{\sqrt{2}Y(t)}{\sqrt{3}I_n}\frac{de_{jk}^{n}}{dt}\left[\frac{A_n\chi_n^2 - 1 - \chi\left(A_n-1\right)}{A_n\chi_n^2+1}\right]} \tag{227}$$

The total strain is then

$$e_{jk}^{T} = \frac{1}{2G}\left(s_{jk}^{1} + 2G\sum_{n=1}^{N}\dot{e}_{jk}^{n}\Delta t^{n}\right)$$

and the total plastic strain can be written as

$$e_{jk}^{P} = \frac{1}{2G}\sum_{n=1}^{N}\mathscr{I}^{n}$$

Thus making the final solution for stress as follows

$$\boxed{s_{jk}^{N+1} = 2G\left(e_{jk}^{T} - e_{jk}^{P}\right)} \tag{228}$$

Notice the equations are very similar to the previous perfectly plastic case, and if flow stress is taken to be constant, these equations reduce to the same Equations shown in section . Notice the only major difference is the coefficient in the first term of $\mathscr{I}$, and the difference in the definition of α.

Fractional Time with Rate-Hardening

As in the perfectly plastic case, the idea of using a fractional time step is needed for correct solutions in the rate-hardening case. However, in this case, the addition of a rate hardening flow stress makes the definition of f, the fractional time coefficient, more complicated. The following plot explains the difference between the equivalent plastic strain depending on the flow stress used.

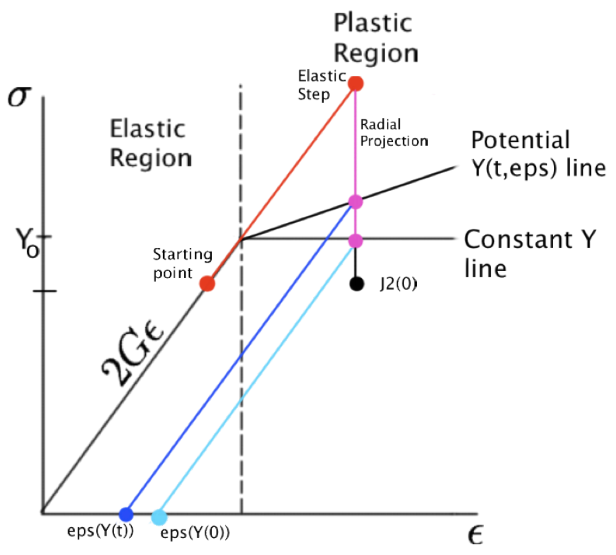


Figure 179: A graphical representation of the error produced when assuming yielded or not yielded during an entire step with rate hardening

The plot shows both a potential rate hardening flow stress stress-strain curve and a constant flow stress curve. Clearly from the plot, the equivalent plastic strain is very different depending on which family of flow stress curves the material follows. Thus, by following a similar reasoning as in section , the fraction of time in a step spent on the yield surface is

$$f = \frac{\sqrt{3J_2^{trial}} - Y(t)}{\sqrt{3J_2^{trial}} - \sqrt{3J_2^{old}}} \tag{229}$$

tThe difference in the rate-hardening fractional time coefficien, f, is the instance of Y(t) versus Y(0) as in the perfectly plastic case. This is a important difference.

Nonlinear Iteration for Rate Hardening Materials

In this section, the need for iterating the solution for stress is introduced. Notice for $\dot{Y} \neq 0$, the final solution for plastic strain is strongly a function of $Y(t)$, and the equation is highly nonlinear in $Y(t)$. This would not be a problem if $Y(t)$ were known during the course of a step. However, the flow stress is dependent on the total deformation and deformation of the material, i.e. flow stress is a function equivalent plastic strain and strain rate, and equivalent plastic strain is a function of flow stress. Furthermore, the fractional time coefficient is a function of final flow stress, so this quantity is also apart of the system of nonlinear equations

$$Y \equiv Y(t, \varepsilon, \dot{\varepsilon}) \tag{230}$$

$$\varepsilon \equiv \varepsilon(t, Y, \dot{Y}, f) \tag{231}$$

$$f \equiv f(Y(t)) \tag{232}$$

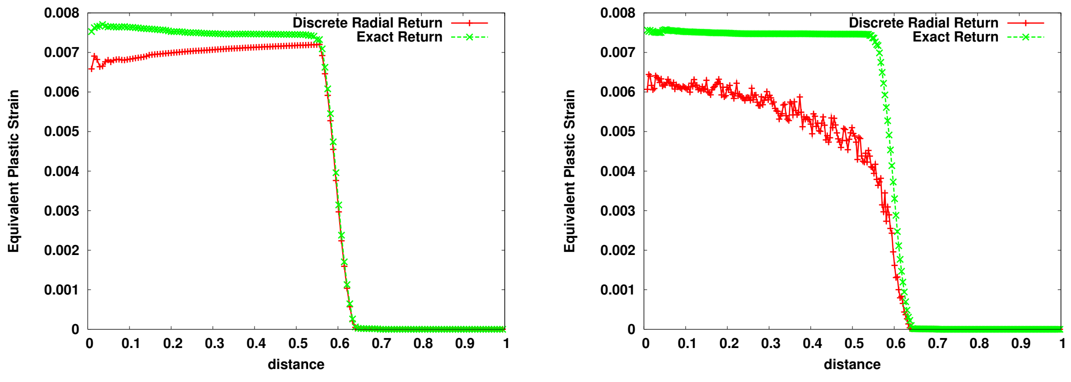
Current implementations of the discrete radial return assumes $Y(t) = Y(0)$, and then calculates the equivalent plastic strain. However, this was shown in Scott Runnels' and Len Margolin's

"An Integrated Study of Numerical Shock Shape, Artificial Viscosity, and Plasticity" [56] to be a likely cause of noise and instability. Therefore, because of the heavy dependence of flow stress and equivalent plastic strain on one another, we decided to implement nonlinear iterations to converge on a solution to both the final flow stress and final equivalent plastic strain. The method used during this report was successive iterations between the two equations. A convergence criteria of the relative error in flow stress and equivalent plastic strain must be less than 1e-10 was used. This criteria in equation form is as follows

$$\left|\frac{Y^N - Y^{N-1}}{Y^N}\right| \leq 1 \times 10^{-10} \tag{233}$$

$$\left|\frac{eps^N - eps^{N-1}}{eps^N}\right| \leq 1 \times 10^{-10} \tag{234}$$

Thus, when both values have converged to within 1e-10, the final solution is accepted as the correct solution. With this implementation, the plots shown in Figure 180 were created.



(a) coarse mesh: 300 grid points

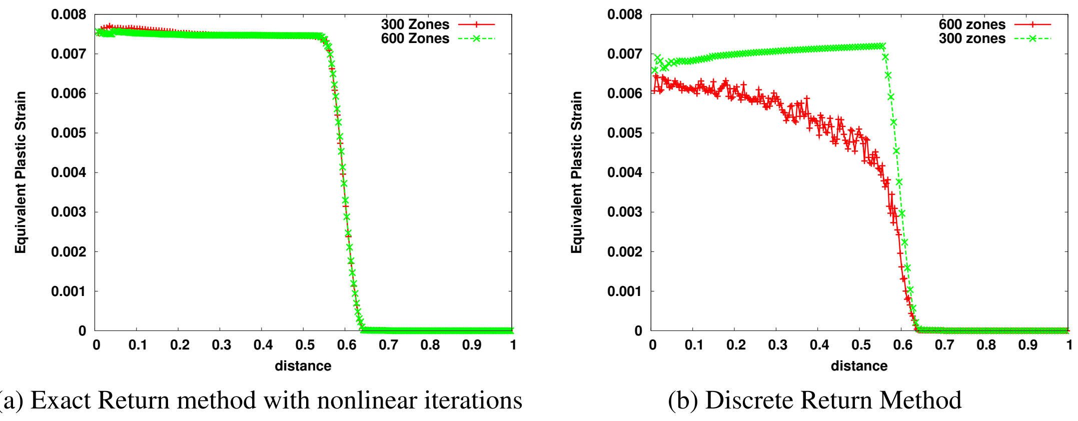
(b) coarse mesh: 600 grid points

Figure 180: Equivalent plastic strain versus distance for 1-D piston problem with rate hardening material (PTW)

These plots show that when the methods are applied to a rate hardening model material, the solutions are very different. The Exact Return produces far more accurate solutions even on this simple 1-D piston problem than the discrete radial return method did. Note, also, that the discrete method was anti-convergent spatially.

Convergence Tests of Iterated Exact Solution

A large portion of this study is due to the need for a more correct solution to the strength model of materials. As shown in Scott Runnels' [56], the current rate hardening model combined with the discrete radial return, without a non-linear iteration scheme, exhibits spatial anticonvergence. The non convergent nature of the current FLAG implementation is a major motivation for this study. Because of the anticonvergence of the FLAG implementation, much work has been devoted to finding a correct solution to the model. Current attempts include creating a more exact analytical solution to the rate-hardening model based on exactly solving the model's governing equations. This attempt is studied and discussed in the Section PTW Model. The second attempt is the exact solution to the radial return algorithm, however, implementing the exact solution to the radial return without non-linear iterations also experienced similar noise and anticonvergence as the discrete radial return method. However, by implementing a non-linear iteration between equivalent plastic strain and flow stress, the method exhibits spatial convergence and stability in the final solution, as shown in Figure 181.



(a) Exact Return method with nonlinear iterations

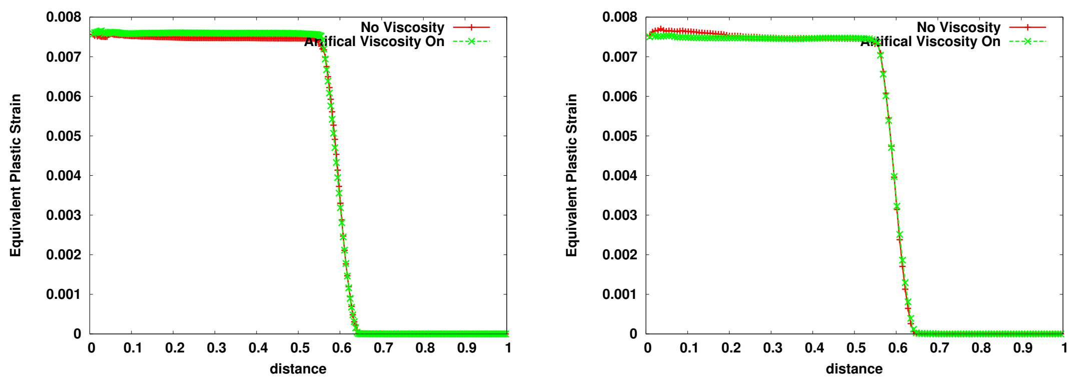
(b) Discrete Return Method

Figure 181: Equivalent plastic strain versus distance for 1-D piston problem with rate hardening material (PTW)

This plot shows that while the Discrete method (right) is not spatially convergent, the Exact method with nonlinear iterations is spatially convergent. This is a huge improvement over the Discrete radial return.

Artificial Viscosity Study

In the currently implemented method of Discrete Radial Return, artificial viscosity is often used to mask some of inaccuracies that occur from the discrete method. The artificial viscosity can be used to create the illusion of spatial convergence and smear some of the errors in the solutions[56]. Artificial viscosity was shown to have a very strong impact on the solution produced by the Discrete Radial Return method in [56]. While the artificial viscosity has a large effect on the Discrete method, artificial viscosity has a much smaller impact on the Exact Method, as shown by Figure 182.



(a) Exact Solution with Fine Mesh, 600 points (b) Exact Solution with Coarse Mesh, 300 points

Figure 182: Equivalent plastic strain versus distance for 1-D piston problem with rate hardening material (PTW). These plots show the effects of artificial viscosity on the Exact Method's solutions

These plots show that the artificial viscosity has very low impact on the final solution of the 1-D piston problem. This is good because this implies that the underlining physics the artificial viscosity was imitating previously is now being implemented through the exact return and nonlinear iterations. This is an improvement over the Discrete method which required artificial viscosity to mask inaccuracies in the solution [56].

PTW Model

In this section, the relationship between strain rate and rate hardening is studied. One model for describing rate hardening is the PTW model, created by Preston, Tonks, and Wallace [50]. This model is shown to be governed by an ODE.

$$\frac{d\hat{\tau}}{d\varepsilon} = \theta \frac{\exp\left[p\frac{\hat{\tau}_s-\hat{\tau}}{s_0-\hat{\tau}_y}\right]-1}{\exp\left[p\frac{\hat{\tau}_s-\hat{\tau}_y}{s_0-\hat{\tau}_y}\right]-1} \tag{235}$$

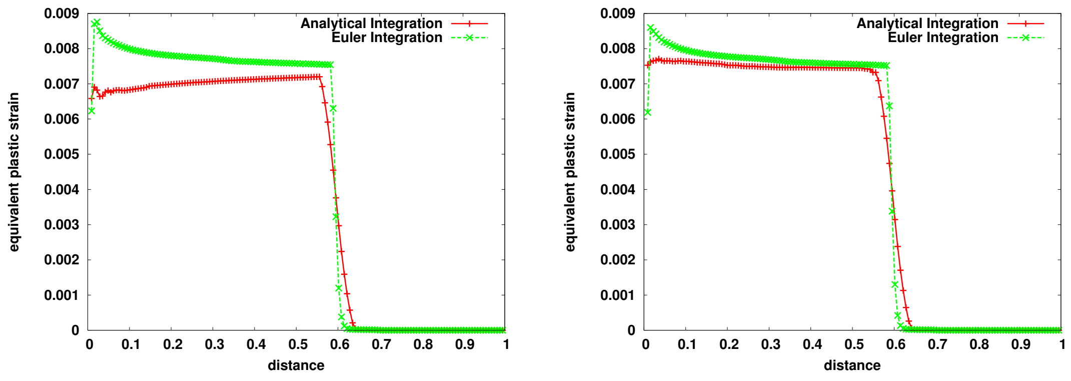
where p is a dimensionless material parameter, $s_0 = \hat{\tau}_s(T=0)$,τ is flow stress, ε is the strain, τ_s is the saturation stress, and τ_y is the yield stress. This section describes the current implementations of this model, and then continues by discussing a new exact implementation of the model.

Current Implementation of PTW

Currently, the PTW model ODE has been integrated by two different methods. The first method, was referred to in [56] as the 'Query Method'. This method integrates the ODE by assuming the strain rate is constant over the entire simulation (note the strain rate is typically assumed constant only during a simulation step), this means that the analytical solution is adjusted at each time step to reflect changes in the saturation and yield stress for the current time step. [50]. Under this assumption the PTW model can be shown to be

$$\hat{\tau} = \hat{\tau}_s + \frac{1}{p}(s_0 - \hat{\tau}_y)\ln\left[1 - \left[1 - \exp\left(p\frac{\hat{\tau}_s - \hat{\tau}_y}{s_0 - \hat{\tau}_y}\right)\right] \times \exp\left\{-\frac{p\theta\psi}{(s_0 - \hat{\tau}_y)\left[\exp\left(p\frac{\hat{\tau}_s - \hat{\tau}_y}{s_0 - \hat{\tau}_y}\right) - 1\right]}\right\}\right] \tag{236}$$

where ψ is the total strain in the simulation. This equation when implemented will be known as the query method, or analytical integration method. The Query method for the PTW Model is currently the default model in FLAG. This model was the one used for all studies presented in the previous sections. Now we will study the other potential methods of solving the PTW Model ODE. The second method is a simple forward Euler integration of the integral with the assumption that the strain rate is constant over a step. When these methods are compared to one another the following plot is created.



(a) Discrete Method (b) Exact Method with Iterations

Figure 183: Equivalent plastic strain versus distance for 1-D piston problem with rate hardening material (PTW). 300 grid points.

From Figure 183 (a), clearly with the discrete method the choice of solution for the PTW ODE can alter the solution drastically. However, the second plot shows that the Exact method with nonlinear iterations makes the difference between the PTW solutions much smaller.

However, both of these methods were considered inaccurate [56]. The issues with the Query method was that the use of curves integrated with the assumption of constant strain rate over the entire simulation with only saturation and yield stress changing temporally is considered an assumption that led to non-convergence. Basically, if there are a family of possible solutions to the ODE, the Query method jumps from one curve to another with no knowledge of the past values or states. This method is discrete, and unreliable. The forward Euler method, while a legitimate method of approximating an integral, was assumed to produce smoother, more convergent results in part because of it diffusiveness. Thus, while the forward Euler integration is commonly implemented, the solution was not considered to be reliable because the solution may have been overly diffusive. This was the motivation for finding a more exact solution to the PTW model ODE.

Exact Solution

Because of the inaccuracies in the current implementation, the PTW ODE was integrated exactly under the assumption that the strain rate was constant over a single step [56]. With this assumption, the ODE is solved to be

$$\hat{\tau}_{n+1} = \hat{\tau}_{sn} + \frac{(s_0 - \hat{\tau}_{yn})}{p} \ln \left[1 - \left(1 - \exp\left[p \frac{\hat{\tau}_{sn} - \hat{\tau}_n}{s_0 - \hat{\tau}_{yn}} \right] \right) \times \exp\left(- \frac{p\theta\Delta\varepsilon_n}{(s_0 - \hat{\tau}_{yn}) \left[\exp\left(p \frac{\hat{\tau}_{sn} - \hat{\tau}_{yn}}{s_0 - \hat{\tau}_{yn}} \right) - 1 \right]} \right) \right] \tag{237}$$

where the addition of a $_n$ represents the value at the beginning of a time step, and $\Delta\varepsilon_n$ is the amount of strain accumulated during the time step. While this is the correct form of the exact equation, the equation had an error in the original print in [56]. This exact integration was found to be just as inaccurate as the Query method when implemented in FLAG in the LA-UR. Further, the method was concluded to be very similar in shape and structure as the Query solution in the LA-UR. However, when the implementation was further investigated, at least two important errors were discovered. The solution was re-implemented for this report.

Exact PTW Results

The Exact Solution was re-implemented into FLAG. When this equation was implemented correctly and applied to a 1-D piston problem with the discrete method applied, the plot in Figure 184 was produced.

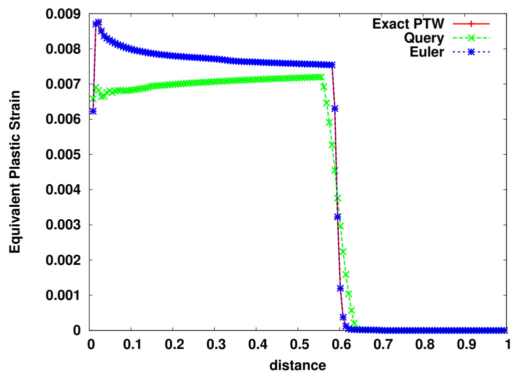
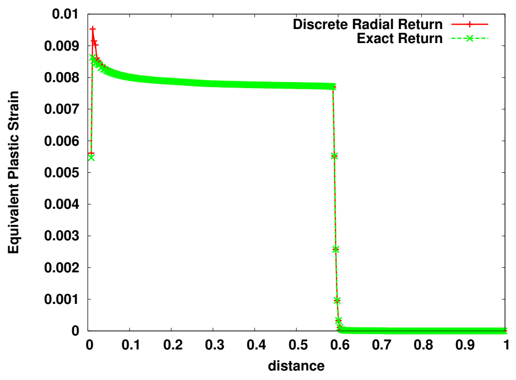
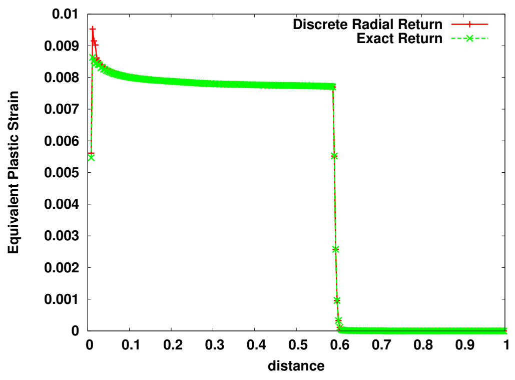


Figure 184: Equivalent plastic strain versus distance for 1-D piston problem with rate hardening material (PTW). This plot was created with the discrete radial return method. There are no nonlinear iteration applied to any of these lines.

This figure shows the equivalent plastic strain versus distance for a 1-D piston problem. The figure shows that contrary to the original report, the exact solution to the PTW ODE is the same point for point to the forward Euler solution. This figure shows that the Query method does indeed produces incorrect results even on a simple problem. However, this figure shows that the forward Euler integration may not be as diffusive or inaccurate as previously thought. The two lines being the same verifies code to code that the Exact PTW solution and Euler integration are implemented correctly. This plot was shown with the discrete method implemented, the next plots show the results with the exact method with nonlinear iterations implemented.



(a) Euler Integration



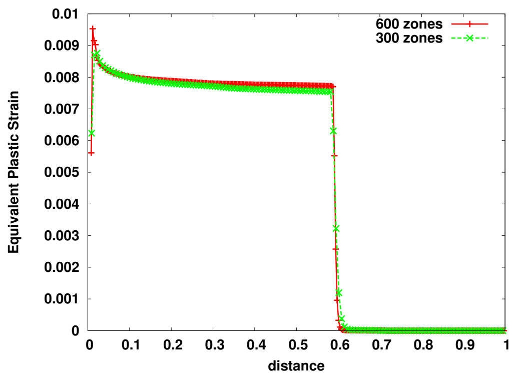
(b) Exact PTW Solution

Figure 185: Equivalent plastic strain versus distance for 1-D piston problem with rate hardening material (PTW). These plots compare the use of the exact method and nonlinear iterations for Euler and Exact PTW solutions

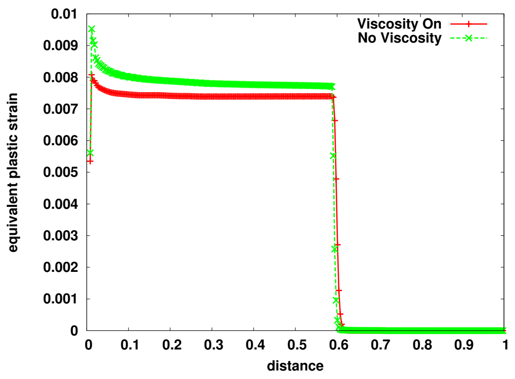
While, the forward Euler method produces similar results to the exact integration, this was a simple 1-D piston problem, and on a more complex shock problem, more issues with the forward Euler method may arise. Because of the lack of reliability in the Euler method, the exact solution to the PTW model should be considered an improvement to the current implementation. Also, an exact equation for PTW allows the rate hardening model to be elevated to the level of the hydro equations.

Additional Studies in PTW

In the section for Rate-Hardening the Query method with the discrete radial return method was shown to be largely impacted by artificial viscosity and spatially nonconvergent. Also in the Rate-Hardening section the Query Method paired with the Exact Return was shown to be convergent and minimally affected by artificial viscosity. In this section we will show similar studies for the Euler method and Exact method for PTW. First, the discrete radial return method was paired with Euler PTW to create the following plots.

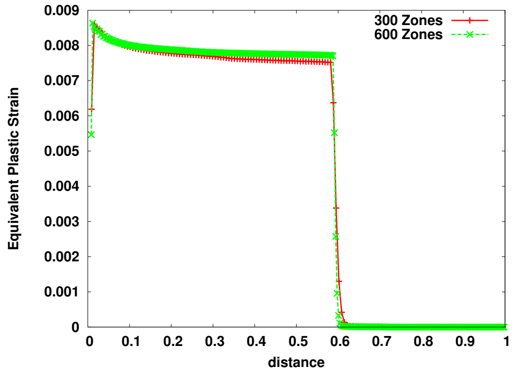


(a) Spatial Convergence Study

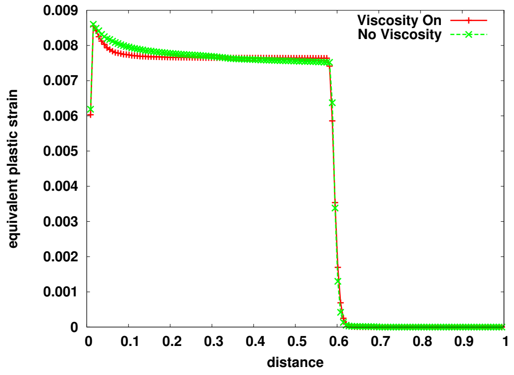


(b) Viscosity Study

These plots show that the Euler method with discrete radial return is spatially convergent, however, the artifical viscosity has a rather large effect on the solution. Next, the study was done on the Exact Return paired with nonlinear iterations and the Euler method.

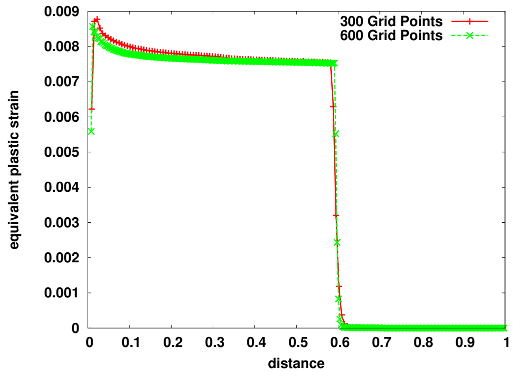


(c) Spatial Convergence Study

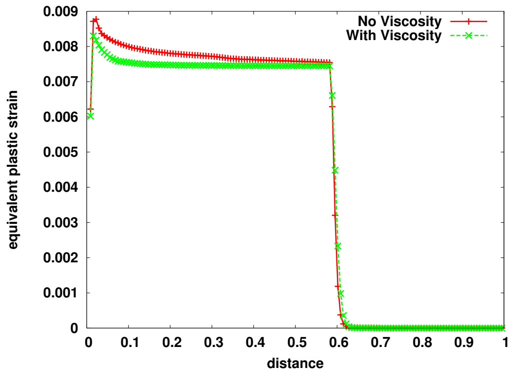


(d) Viscosity Study

These plots show that the Euler method with Exact return and nonlinear iterations experiences spatial convergence and very little effects from artificial viscosity. Next the study was done on the Exact Return method paired with Exact PTW but without nonlinear iterations.



(e) Spatial Convergence Study



(f) Viscosity Study

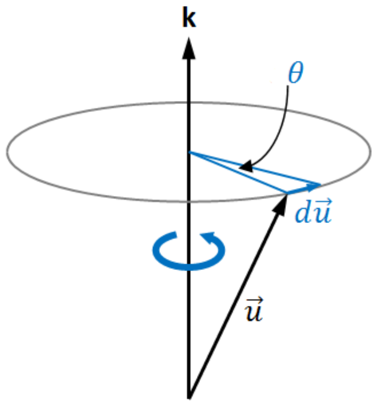
These plots show that the Exact Return with Exact PTW is spatially convergent and has less impact from artificial viscosity than Discrete Return with Euler PTW but more impact than Exact Return with Euler PTW and nonlinear iterations. This implies that the Exact PTW is less sensitive than the Euler PTW when artificial is used. These figures also imply that if nonlinear iterations were used with Exact Return and Exact PTW, the effects of artificial viscosity would be nearly non existent. The conclusions of the 1-D piston problem are summarized in the following table.

Test Case	Exact Return	Discrete Radial Return	Nonlinear Iterations	PTW: Query	PTW: Euler	PTW: Exact	Spatially Convergent	Viscosity effects	Notes
1		X		X			No	Drastically effects solution. Viscosity smears solution, masks inaccuracies and nonconvergence	Solutions are very inaccurate
2		X			X		Yes	Viscosity smears the solution, wall heating is smeared	Large wall heating present in solution
3	X		X	X			Yes	Minimal effect, solution smears but not by a lot	Solution is closer to test case 2 solution, but still not exactly the same
4	X		X		X		Yes	Minimal effects, solution wall heating is smeared	Solution is very similar to test case 2
5	X					X	Yes	Minimal effects, solution wall heating is smeared	Solution is identical to test case 2. No nonlinear iterations was implemented in combination with PTW:Exact

Dealing with Finite Rotations

Modeling Finite Rotations with the Rodrigues' Vector

Finite body rotations of a material can be modeled as an angular rotation about some axis. The axis of rotation is a unit vector called the Rodrigues vector ($\mathbf{k}$), and the accompanying angle is the Rodrigues angle (θ). From these two variables, one can model the rotation of any vector or tensor about that axis in multiple dimensions. In order to directly calculate the rotation of a desired vector or tensor, one can derive the full rotational tensor ($\mathbf{R}_{\mathrm{ij}}$) using the Rodrigues formula. more information on strain and deformation with finite rotations, see [20] and [38].



Deriving the Rodrigues Formula

We start with simple vector notation for the rotation of the vector ($\mathbf{v}$) about the axis ($\mathbf{k}$) by an angle (θ)

$$\mathbf{v}_{\mathrm{rot}} = \mathbf{v}\cos(\theta) + (\mathbf{k}\times\mathbf{v})\sin(\theta) + \mathbf{k}(\mathbf{k}\cdot\mathbf{v})(1-\cos(\theta)) \tag{238}$$

In matrix notation, this formula is

$$\mathbf{v}_{\mathrm{rot}} = \mathbf{v}\cos(\theta) + [\mathbf{k}]_{\mathrm{x}}\mathbf{v}\sin(\theta) + \mathbf{k}\mathbf{k}^{\mathrm{T}}\mathbf{v}(1-\cos(\theta)) \tag{239}$$

Where $[\mathbf{k}]_{\mathrm{x}}$, the cross-product matrix for $\mathbf{k}$ is

$$[\mathbf{k}]_{\mathrm{x}} = \begin{bmatrix} 0 & -k_3 & k_2 \\ k_3 & 0 & -k_1 \\ -k_2 & k_1 & 0 \end{bmatrix} \tag{240}$$

With some manipulation, the rotated vector can be expressed as

$$\mathbf{v}_{\mathrm{rot}} = (I\cos(\theta) + [\mathbf{k}]_{\mathrm{x}}\sin(\theta) + \mathbf{k}\mathbf{k}^{\mathrm{T}}(1-\cos(\theta)))\mathbf{v} \tag{241}$$

Therefore, $\mathbf{v}_{\mathrm{rot}}$ can be expressed as a $R\mathbf{v}$ where

$$R = I\cos(\theta) + [\mathbf{k}]_{\mathrm{x}}\sin(\theta) + \mathbf{k}\mathbf{k}^{\mathrm{T}}(1-\cos(\theta)) \tag{242}$$

This is equivalent to

$$\boxed{R = I + [\mathbf{k}]_{\mathrm{x}}\sin(\theta) + (\mathbf{k}\mathbf{k}^{\mathrm{T}} - I)(1-\cos(\theta))} \tag{243}$$

Using the Rotational Tensor

Now we have the final form of the rotational tensor which we can use to handle finite rotations. The main use of R is rotating vectors or tensors between the material frame (the material's actual orientation) and the lab frame (the material's original orientation). The equation to rotate the tensor σ_{ij} from the material frame to the lab frame is

$$\boxed{\sigma_{\mathrm{ij}}^{lab} = R_{\mathrm{ij}}^{\mathrm{T}}\sigma_{\mathrm{jk}}^{\mathrm{mat}}R_{\mathrm{kl}}} \tag{244}$$

And when rotating the tensor from the lab frame to the material frame, one must use

$$\boxed{\sigma_{\mathrm{ij}}^{mat} = R_{\mathrm{ij}}\sigma_{\mathrm{jk}}^{\mathrm{lab}}R_{\mathrm{kl}}^{\mathrm{T}}} \tag{245}$$

In order to account for rotations when using the Exact Return method one can update total strain and total plastic strain within the lab by rotating strain rate and deviatoric stress into the lab frame before calculating the strains of the new time step and rotating strain rate and deviatoric stress back into the material frame afterwards. When considering advection, the total stress must be rotated into the material frame after being advected using the strains which are constantly in the lab frame.

Observing the Sensitivity to Relaxers in ALE

A mesh relaxer in FLAG is what determines how the mesh moves during the advection step in ALE. It has been observed in the original version of FLAG that changing the mesh relaxer can change the results. It is disturbing that one can arbitrarily choose a mesh relaxer and arrive at a different conclusion than someone who used a different relaxer. Therefore, it was our goal that the new methods would mitigate this sensitivity to relaxers used in ALE.

Results

We tested our handling of rotations on the Taylor Anvil problem which is a solid metal cylinder slamming against a rigid surface.

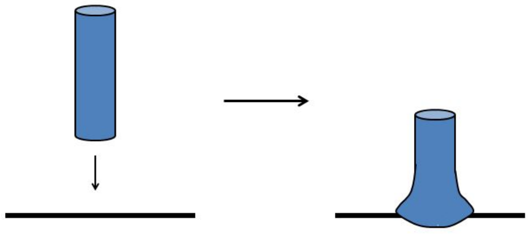
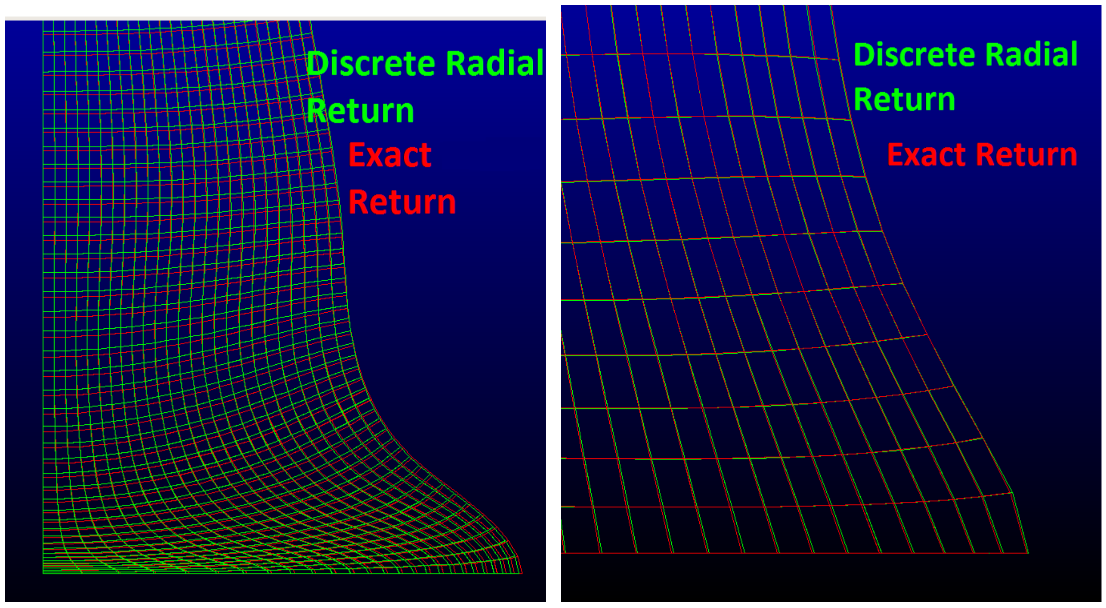


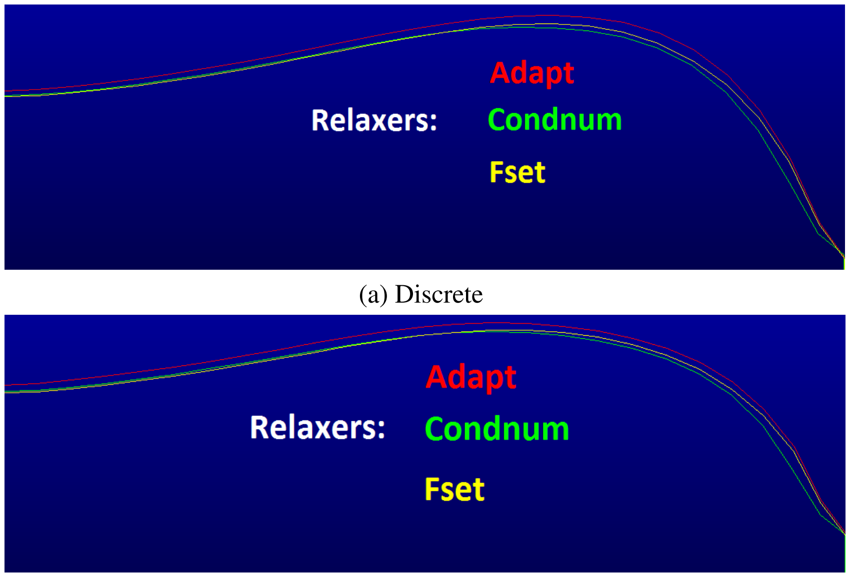
Figure 186: This is an illustration of the Taylor Anvil problem. We are interested in the outer shape of the foot.



(a) Lagrange (b) ALE

Figure 187: These two plots compare the Exact Return method and the Discrete Radial Return method in a Taylor Anvil problem. The left is in Lagrange, and the right is in ALE

As you can see from these graphs, the outer shape of the anvil matches very closely for the Exact Return and Radial Return algorithms in both Lagrange and ALE. This verifies that we handled finite body rotations correctly as we arrived at similar results. We would not expect identical results because the calculations for both methods are quite different. This assures us that the Exact Return method works not only in one dimension but also in multiple dimensions.



(a) Discrete

(b) Exact

Figure 188: These two plots compare the Equivalent Plastic Strain along the foot of the Taylor Anvil for three different relaxers: adapt, condnum, and fset. The left is the Discrete Radial Return, and the right is the Exact Return

As you can see from these figures, each relaxer had about the same effect on the equivalent plastic strain at the foot of the anvil. Moreover, each relaxer had a nearly identical effect on the outer shape of the anvil. Therefore, the Exact Return method coupled with the Rodrigues rotations did not correct the sensitivity to relaxers in ALE as we were hoping.

Conclusions

Considerations on the Methods and Future Work

One important consideration must be made in the calculation of α in the rate hardening study. In that section, the flow stress was considered linear in time in order to evaluate the integral of α. The actual impact of this assumption was not studied, but very well could be a poor assumption. However, one possible solution is to use the solution of the rate hardening model for flow stress and evaluate the integral with this equation (if the integral is evaluatable). Another potential solution is to use a leap frog method to acquire more data points for flow stress, and then use quadrature approximation of the integral. Several others exist, if this is determined to be a large source of errors.

Another important consideration must be made in the derivation of f for the rate hardening case. The derivation of f was done entirely through the geometry of Figure 179. However, when a material experiences rate hardening, the moduli of the material are no longer constant. If the moduli change, particularly, the shear modulus, the geometry of Figure 179 will change slightly. The elastic unloading lines may no longer be in parallel with one another, which would alter the value of f slightly. The solution to this issue is a slightly different equation for

f, the form would generally be the same, but would be more complicated.

A third consideration must be made for the nonlinear iterations. The method of root finding in this report for the nonlinear system of equations in flow stress and strain and strain rate was successive iterations. This method was chosen for its simplicity and reliability. However, with this report showing the success of both an exact equation for strain/strain rate and for the PTW rate hardening model, more advanced options for root finding are possible. One could implement Secant method or Newton's method of root finding in order to create a faster converging routine than the one used in this report.

A fourth consideration must be made for the method of advecting the rodrigues vector. Currently in FLAG, the Rodrigues' vector is volume weighted during advection then scaled to ensure that it has a magnitude of one. Similarly, the sine and cosine of the Rodrigues' angle are volume weighted then scaled to ensure that $\cos(\theta)^2 + \sin(\theta)^2 = 1$. This way of advecting the Rodrigues' vector and angle is non-physical, and thus raises questions as to the validity of this approach. One possible avenue of investigation would be weighting the Rodrigues' vector based on the Rodrigues' angle. This would be a better approximation to the rotation of a zone. It may also be more accurate to volume weight the Rodrigues' angle itself rather than its sine and cosine. Further investigations into the advection of these variables would be highly beneficial.

A final consideration is made for the stability of the equations used. While no explicit temporal studies were done during the scope of this report, the authors did complete a couple of tests runs to study the effects of the time step used. While changing the time step used for the simulations was capable of changing the solution with the discrete method, the new method was much less sensitive to time step changes. The majority of this can be attributed to the discrete nature and non iterative methods of the old method being dependent on the time step used where as the new methods are exactly solved and iterated making the solutions less sensitivity and more stable.

Conclusions

This report has been a multifaceted study of strength modeling in hydro codes. First, some of the issues with the current method of modeling the plastic regime was shown. Next, the derivation of a strain-based method for modeling the plastic regime was shown. With this strain-based method, the method was verified by code-to-code comparison in the production code FLAG. The method was tested on a perfectly plastic material and shown to produce accurate solutions. With the method verified the study was expanded to multi-D, rate hardening, and exactly solving the rate hardening model. The strain based method was extended to be capable of handling rate hardening materials. By introducing nonlinear iterations, the method found convergent and accurate solutions for the rate hardening material. Next, the use of an exact solution to the PTW model ODE was studied and shown to be the same solution as forward Euler integration. Last, the ability to use the method in multi-D was studied. Overall, the report shows the new strain based method is accurate, spatially convergent, more physical in advection, capable of handling multi-D, and computationally efficient.

Notes For the Reader

Note, that in both papers, the J_2 invariant restriction is as follows

$$J_2 \leq Y^2$$

instead of the FLAG used version of

$$J_2 \leq \frac{Y^2}{3}$$

thus the derivations included in this report differ slightly from the original derivations in [39] and [40].

Also to be noted, each integral is taken to be over a step in the simulation. Thus 0 represents the beginning of a step and t represents the time at the end of the step.

For in depth derivations for the PTW solutions, the Query method can be found in [50] and the Exact solution can be found in [56]. The derivations were not included in this report because the analysis on PTW rate hardening model was done at the end of the summer. However, the plots and implications of the exact method were included in the report.

Note that to truly see the effects of ALE versus lagrange mode, the piston problems in section were run with a much faster piston speed than in the piston problems used in the rest of the report.

Acknoledgments

The authors would like to acknowledge Len Margolin for creating the Exact Return method. The authors would like to acknowledge the LANL Computational Summer Workshop, which provided us with the opportunity to pursue this problem. Mostly, the authors would like to acknowledge Scott Runnels for being a great mentor, with out him certainly we would have been fish out of water.

Implementation into FLAG

Alterations to MatModel/Mat/SolidPiece.dic class SVPBase

Three variables added var environment:

```
    variable initializer “used to initialize het
        (TYPE i_0 INIT 0 UNITS none)

    variable het “Total Elastic Strain”
        (TYPE r_6H_v INIT 0.0 UNITS none)

    variable hept “Total Plastic Strain”
        (TYPE r_6H_v INIT 0.0 UNITS none)
```

initializer is used in ExactODE.g. It is initalized as 0, thus, in ExactODE.g, if the variable is 0 then het is initialized as the total strain divided by 2G. Then initialized is set to 1.

Two existing variables are altered:

  variable hs
    PreAdvect, ADVECT, AggSizeADVECT, AggPreADVECT, AggPostADVECT attributes removed
    change responder to broadcast PostAdvect from PostAdvHS to PostAdvHSODE

  variable hs0
    PreAdvect, ADVECT, AggSizeADVECT, AggPreADVECT, AggPostADVECT, PostAdvect attributes removed

Alterations to DELFI/MeshOpt/adv/postadvect.g
addition of subroutine PostAdvHSODE(hvar, advection)

Alterations to MatModel/Solid/Decoupled.g, remove access of tauh0 if isocgy
additional access variables:
  /kdd/ initializer
  /kdd/ strength, /strength/ ptwmod1, /ptwmod1/ tauh0(kkhll)
  /kdd/ het(6, kkhll)

tmp and sd0tmp.
Called mat2lab twice before the addition of variables:
  real*8 tauh_temp(kkhll)

tauh0 and tauh_temp were used around the SBroad of 'Strength' to reset tauh0 after the strength call

SBroad of 'Elastic Stress' was removed

Broadcast for 'PlasticFlow', and call for IsoPlasticAF was removed.

After broadcast for PlasticFlow initialization of het was added
Broadcast for ODEStress and call to ExactStressODE(kdd,kkhl,klist,s_d0,S_D,dt,hg,hhee,hyf) was added

We implemented the code to always use ExactStressODE by default, however, a class could easily be made such that when instantiated, ExactStressODE responds to plasticflow, making the subroutine fit into the standard of FLAG.

Alterations to MatModel/Constitutive/Yield/PTWMod1.dic
Variables added to class PTWMod1
  variable tauh_y_n "Yield stress from previous cycle"

(TYPE r_H_v INIT 0. UNITS pressure)

variable tauh_s_n "Saturation stress from previous cycle"
(TYPE r_H_v INIT 0. UNITS pressure)

variable tauh_n "Flow stress from previous cycle"
(TYPE r_H_v INIT 0. UNITS pressure)

Alterations to MatModel/constitutive/Yield/PTWMod1.g
Any line pertaining to the new variables with _n has been added to the file. Most likely the easiest way to handle this is to search for _n and add those lines. Also all lines with integrate_tau == 2 if statement has been added for calculating tauh

Addition of file MatModel/Solid/ExactODE.g
This file is where all the meat of the Exact Radial Return is

For Rotations

Alterations to /MatModel/Solid/Decoupled.g
Added an argument, rot, to the ODEStress call in order to pass the rotational tensor.

Alterations to /MatModel/Solid/ExactODE.g
Added rot to the arguments for the subroutine.
Created temporary variables hheetmp and sd0tmp.
Called mat2lab twice before the calculations
Called lab2mat once after the calculations.

Alterations to /DELFI/MeshOpt/adv/postadvect.g
Accessed hrodv, hcosr, and hsinr in subroutine PostAdvODEHS.
Called Rod2R at beginning of that subroutine.
Called lab2mat once at the end of the subroutine.

Next Steps
If one were to want to iterate with the exact integration of PTW (i.e. integrate_tau==2) then one would need to add statements similar to the tauh0 statements in Decoupled.g and ExactODE.g except for variables tauh_n, tauh_s_n, and tauh_y_n. Such that all four variables are reset after the calls for strength except for the final call to strength in a step.

libparty: An Efficient Particle Library For Advanced Architectures

By Matthew Kinsey and Verinder Rana, Jon Reisner (mentor), Robert Robey (mentor), Jeremy Sauer (mentor)

Abstract

As the scale of physical simulations moves to exascale and beyond, the use of highly parallel algorithms is becoming increasingly important. Future hardware appears to be following a trend where improvements in the clock, memory transfer and DRAM latency rates have stagnated relative to the overall number of computational cores available. If parallelized with care, many particle methods fit naturally into this 'new' approach at scientific computation.

In this work, we present `libparty`, a library for large-scale particle simulations targeted specifically at this highly parallel model of computation. This library implements a number of common algorithms and data structures relevant to parallel particle simulation. Currently, the library's methods are being targeted to scale well with a large number of Intel's Phi coprocessors and Knight's Landing processors via both OpenMP and MPI. In this paper, we will discuss various design choices that have enabled `libparty` to achieve excellent scalability with common particle methods in the quest for optimization on these new architectures.

Introduction

The Intel Many Integrated Core (MIC) Architecture

The Intel Many Integrated Core (Intel MIC) architecture is Intel's latest design targeted for processing highly parallel workloads. These chips, referred to as accelerators, are designed to compete with GPUs in off-CPU parallel computing capabilities. Intel combines a large number of cores into one chip so users must write code that is scalable to a large number of concurrent tasks in order to maximize the capabilities of the chip's design. The current generation of Intel MIC cards, codenamed Knights Corner, provide up to 61 cores (244 threads), each with 512 bit-wide SIMD registers, on a single chip. With these cores operating at 1.24GHz, these 244 concurrent threads are able to produce an impressive 1.2 teraFLOPS.

However, the primary problem with the current generation of MIC is that they operate through the PCIe bus, making memory operations between host and accelerator painfully slow. Luckily, there are plans to remedy this issue with the next generation of MICs, codenamed

Knight's Landing. Knight's Landing chips will be hosted on the same die as the main CPU, hopefully speeding up memory operations dramatically (and potentially introducing shared memory spaces). This will likely make the Intel MIC architecture a very attractive component for traditional high performance computing (HPC) environments.

As a result of the highly parallel nature of these devices, algorithms must be able to minimize the amount of serial execution and memory transfers in order to see a performance gain. While many traditional grid based methods may be difficult to conform to these requirements, particle based methods naturally fit within this programming paradigm. As a result, this work is focused on the use of such methods on these new architectures.

Particle Methods

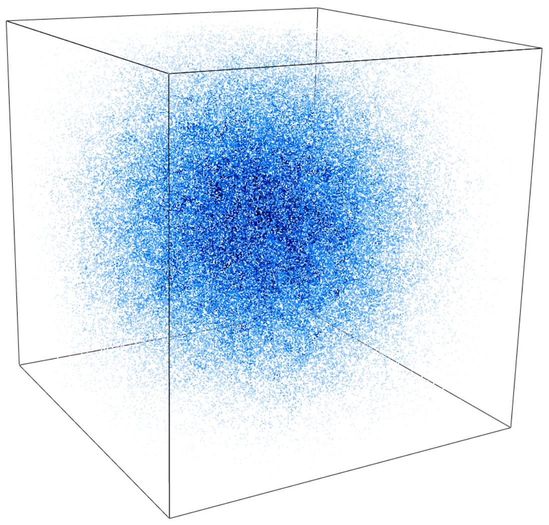


Figure 189: Calculated density of a particle distribution in `libparty`

Particle methods[4] refer to a class of simulations involving the motion of a large number of potentially interacting particles. In these simulations, simple particles carry properties of the physical system, acting as a mesh of sorts while being moved on their trajectories through the physical domain. Particle methods enjoy wide usage in a large range of physical simulations, from modeling the structure of a tiny protein to simulating the cosmological structure of the universe created by perturbations fractions of a second after the Big Bang. While, at first glance, these methods appear to be a deceptively simple, great care must be taken during implementation as poor choices can easily lead to poor performance.

Even while such methods are employed in a diverse set of applications, there are many common algorithms that are at the heart of most, if not all, of them. These algorithms can be separated into two categories, those that are completely independent of the other particles and

[4]For example, Particle-in-Cell (PIC), Smoothed Particle Hydrodynamics (SPH) and Molecular Dynamics (MD) methods

those that are not. While it is often trivially easy to naively parallelize the former, one must use great care when attempting the same with the latter.

We opted to investigate the usage and scalability of two of the most common algorithms employed by particle methods; one that is inherently independent and one that is not. Easily the most common and easily parallelizable of particle operations is that of simple advection. Another commonly used algorithm in particle applications is the finding of N nearest neighbors to an arbitrary position in physical space (often another particle). While the naive implementation of such a search is $\mathcal{O}(N^2)$, finding the N-nearest neighbors to a point can be performed in $\mathcal{O}(NlogN)$ using either a kD-tree or a hash based method. Of these two methods, the hash based method is far more parallelizable and thus the one we explored.

libparty

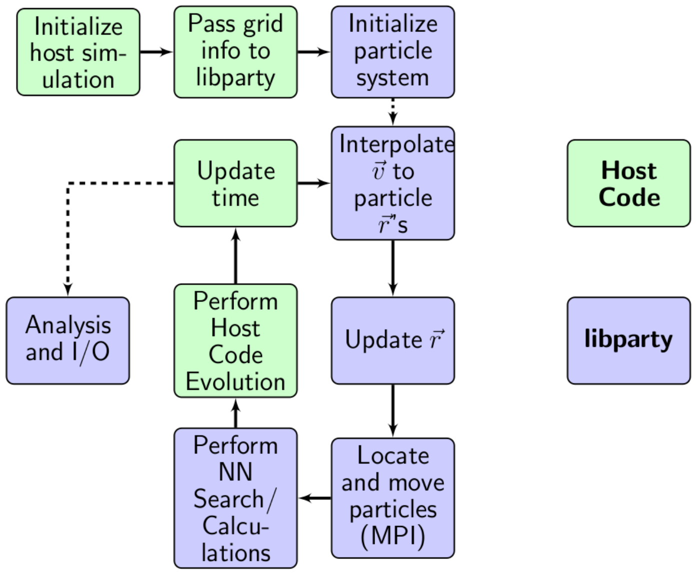

Initialize host simulation
Pass grid info to libparty
Initialize particle system
Update time
Interpolate $\vec{v}$ to particle $\vec{r}$'s
Host Code
Analysis and I/O
Perform Host Code Evolution
Update $\vec{r}$
libparty
Perform NN Search/ Calculations
Locate and move particles (MPI)

Figure 190: Workflow of a typical simulation using libparty

In order to address the challenges this new program model presents in a generic way, we have developed a library containing highly scalable algorithms for particle applications on future architectures, dubbed libparty. To be of use in HPC applications, it is important that libparty be callable from a variety of programming languages. As such, the library contains bindings for Fortran 90 and ANSI C, in addition to its native C++.

Figure 190 shows the division of labor between the host code and libparty for a standard use case. At initialization, libparty gives the host code pointers to its memory space, allowing for seamless access to the internal workings of the library. After initialization, the host is free to call the efficient common particle routines implemented in libparty and simply perform it's own operations on the particle data when needed. Figure 191 shows an example usage scenario written in C++.

```
#include "libparty.hh"
...
int npart = pow(10,6);
part_sys parts = part_sys(npart);

<Initialize Host Grid>

for(int i = 0; i < npart; ++i) {
  parts.x[i] = ...; parts.y[i] = ...; parts.z[i] = ...;
}

parts.grid.xmin = xmin;
parts.grid.ymin = ymin;
parts.grid.zmin = zmin;

parts.grid.nx = nx;
parts.grid.ny = ny;
parts.grid.nz = nz;

parts.grid.dx = dx;
parts.grid.dy = dy;
parts.grid.dz = dz;

for(; t < tmax; t+=dt) {
  <Host Evolution>
  parts.interp3dvelwithgrid(&grid_vx[0], \
                            &grid_vy[0], \
                            &grid_vz[0]);
  parts.step(dt);
  parts.mpi_sync();
  parts.calc_density();  // This calls the NN search
  parts.output_h5part();
}
```

Figure 191: Example libparty usage in C++

At it's heart, the library contains data structures to hold an arbitrary number of particles. Potentially more important than the implementation of the particle algorithms, the underlying memory layout of libparty can have profound performance impacts.

Memory Layouts for Particle Data

Array of Structs (AOS)	Struct of Arrays (SOA)	Array of Structs of Arrays (AOSOA)
`struct part`	`struct part`	`struct part`
`{`	`{`	`{`
`  double x, \`	`  double x[1024], \`	`  double x[16], \`
`         y, \`	`         y[1024], \`	`         y[16], \`
`         z;`	`         z[1024];`	`         z[16];`
`};`	`};`	`};`
`struct part \`	`struct part PARTS;`	`struct part PARTS[64];`
`    PARTS[1024];`		
• Cache locality during interactions	• Vectorizable by the compiler	• Naturally vectorized
• Simple implementation	• Simple implementation	• Complex implementation
• Not vectorized by compiler	• No cache locality during interactions	

Figure 192: The three data layouts available in `libparty`

In an effort to explore the impact of different memory layouts, `libparty` implements three different memory layouts for the internal particle data. Examples of these three memory layouts as well as their pros and cons are shown in Figure 192. After much experimentation we found SOA to have the best combination of speed and simplicity for our needs.

Implemented Algorithms

The methods in `libparty` have been tested to ensure that they scale as expected with respect to number of particles, and that they minimize serial execution. In order to assess the degree to which a method is parallel we fit the results of our tests to Amdahl's law.

Amdahl's Law is a model for the expected speedup obtained from parallelizing an algorithm [1].

Given:

- $N \in \mathbb{Z}^+$ is the number of threads being executed,
- $B \in [0,1]$ is the fraction of the algorithm that is strictly serial,
- The time $t(N)$ an algorithm takes to finish when being executed on N threads corresponds to:

$$t(N) = t(1)\left(B + \frac{1}{N}(1-B)\right) \tag{246}$$

Advection Particle advection is a relatively simple operation to parallelize. It simply consists of solving $2N$ independent ODEs.

$$\frac{dx_i}{dt} = v_i \tag{247}$$

$$\frac{dv_i}{dt} = F_i \tag{248}$$

To minimize the overhead incurred transferring data through the PCIe bus, libparty provides the option of sending the grid data to be used in calculating the force on particles instead. libparty then implements a highly parallel trilinear interpolation on the grid to be used in finding the forces. This allows the number of particles in a simulation to scale without increasing the amount of time spent waiting for the PCIe bus.

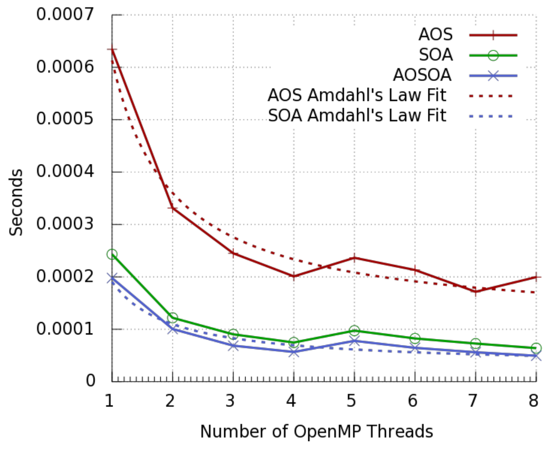


Figure 193: Advection Scaling w.r.t. OpenMP Threads

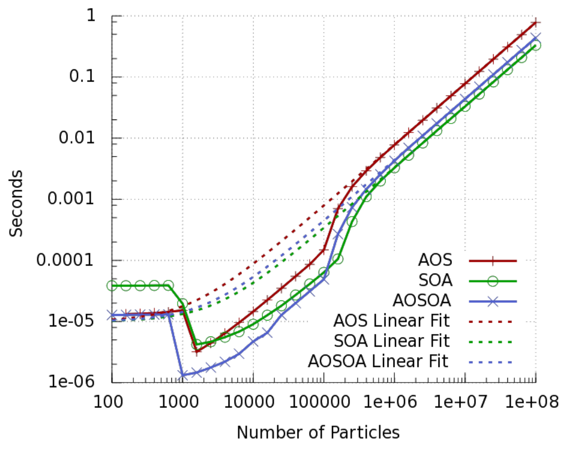


Figure 194: Advection Scaling w.r.t. Number of Particles

N-nearest Neighbor Search libparty implements a Locality Sensitive Hashing (LSH) method to perform N-nearest neighbor searches.

The hash function that maps a position $\vec{x} = (x, y, z)$ to a hash table of size N has the following form:

$$h = \left[\left(\left\lfloor\frac{x}{d}\right\rfloor \cdot p_1\right) \oplus \left(\left\lfloor\frac{y}{d}\right\rfloor \cdot p_2\right) \oplus \left(\left\lfloor\frac{z}{d}\right\rfloor \cdot p_3\right)\right] \% m \tag{249}$$

with p_1, p_2, p_3 being large prime numbers that are chosen to be 73856093, 19349663 and 83492791, m is chosen to be the next prime following $2N$ and d acts as a bin size. The calculation of the hash is completely parallelizable and as a result the overall method is able to scale to large numbers of threads [69].

The hash table is stored in an Intel Thread Building Blocks multimap that allows for fast concurrent accesses which minimize locking.

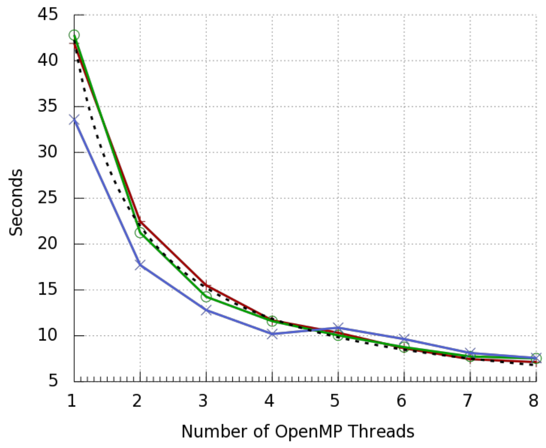


Figure 195: NN Scaling w.r.t. OpenMP Threads

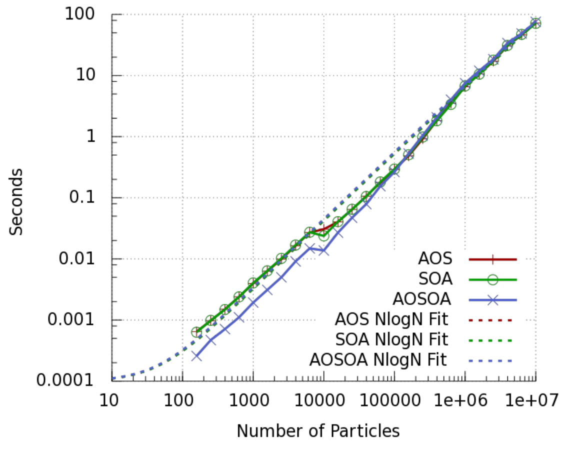


Figure 196: NN Scaling w.r.t. Number of Particles

When we fit the results of the scaling test to Amdahl's Law, we find the the best fits to be $t(1) = 42.2535s$ and $B = 0.04095$. This implies that for 1,000,000 particles approximately 95.9% of the algorithm is performed in parallel.

HIGRAD

`libparty` has seen its first science application in the LANL code, HIGRAD. HIGRAD is a parallel Eulerian atmospheric hydrodynamic framework used for the modeling of wildfire, wind turbines, hurricanes and coupled atmosphere/biosphere dynamics [55]. We simply used `libparty` to perform the time integration of particle trajectories using velocities that `libparty` was able to directly interpolate from the mesh-based hydrodynamics. Beyond initalizing the particles and passing libparty the grid information, we leave all necessary HIGRAD initialization parameters at their default values.

The simulations we performed mimicked a simplified real-world application within cloud physics. Since, saturation vapor pressure and temperature are proportionally related, as we raise the kinetic energy of a group of particles we are able to induce evaporation. In this scenario the parts of the cloud with this higher energy get lighter and rise. `libparty` acted as a stand in replacement for HIGRAD's native particle representation.

HIGRAD Results

We ran a simple Particle-in-Cell simulation when libparty was interfaced with HIGRAD. The simulation was run using four Intel i7-4770K CPUs clocked at 3.5Ghz, compiled using Intel's compiler suite. HIGRAD initializes a uniform three dimensional grid with 200^3 cells in the physical domain. Within the domain, a spherical region was initialized at a higher tempurature than background. As a result, this sphere would deform and rise. In order to test the scaling of `libparty` within this production environment, we initialized an increasing number of particles within each cell, ranging from 1 to 100 particles per grid cell. Figure 197 reports the average wallclock time that our advection algorithm with respect to the total number of particles being simulated.

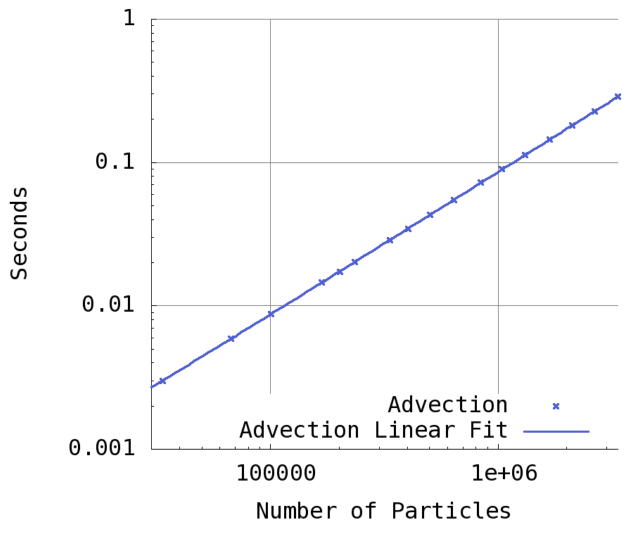


Figure 197: Advection Scaling w.r.t. Number of Particles

Future Work

In its current state, `libparty` is able to interface with a number of scientific codes but there is still much work to be done to accommodate other codes. As a result, there are a number of directions currently being perused to make `libparty` more widely usable. We are currently finishing the implementation of a generic Runge-Kutta integrator that can be performed concurrently with the host code. Another direction currently being presued is a `libparty` method for generic collision operators. Coupled with the nearest neighbor search, collision operators would allow the user to merge or split particles easily, all while `libparty` performs the operation in a highly parallel form.

Perhaps the most interesting open question is that of load balancing. If particles are not distributed uniformly across MPI ranks, when performing particle operations all the threads will be locked for as long as it takes the longest thread to execute. As these uneven distributions are very common in particle applications, there is alot of potential speed to be gained by letting the idle threads help the overburdened threads. To address this issue, we are exploring the use of `libquo` [26], a library that will dynamically change CPU affinities and allow us to let threads execute on CPUs other then the one they spawned on.

We also intend to investigate the usage of `libparty` on Intel's second generation MIC, Knight's Landing. These new chips will alleviate the concerns associated with memory transfers through the PCIe as they are hosted on the same die as the CPU itself. This should allow us to focus on using even larger amounts of parallelism without having to worry about the communication overhead.

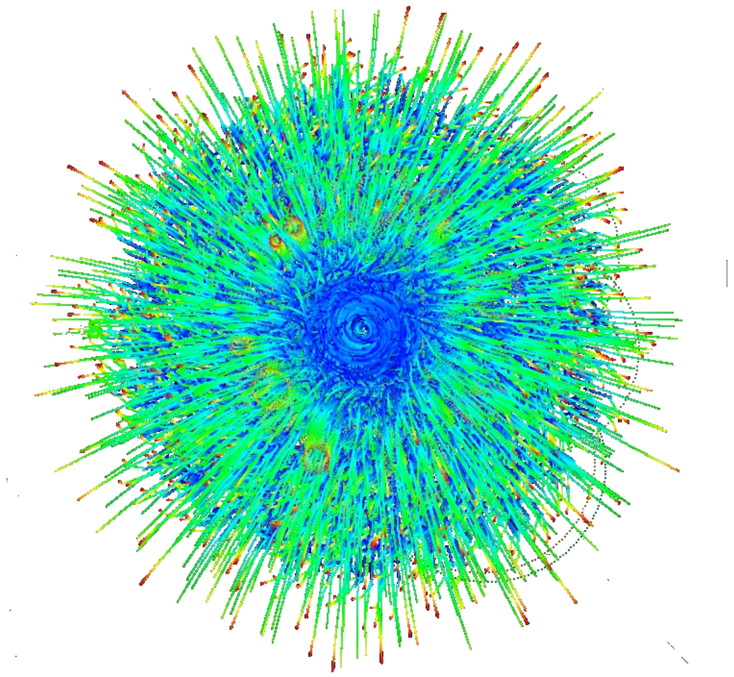


Figure 198: A simulation of charged particles in a black hole jet performed with libparty

Conclusion

As expected, particle methods are natural candidates for use on highly parallel future architectures. In this work, we have investigated how to effectively implement some common particle methods within this highly parallel programming model. We were able to achieve the high levels of parallelism needed to make use of these new architectures. Through the liberal use of both threading and vectorization, this library is able to provide particle methods that are nearly optimally tuned for these new architectures.

libparty is able to provide access to these parallel methods through a very simple API, allowing host code to be easily ported to use the library (e.g. Figure 191). As a result, we were able to use this simple interface to speedup the particle-in-cell methods within HIGRAD, in under 40 lines of additional code.

Acknowledgements

We would like to thank our mentors, Jon Reisner, Bob Robey, and Jeremy Sauer, for providing us with an interesting and important project. Additionally, we would like to thank Scott Runnels and the Computational Physics Student Summer Workshop for providing us the opportunity to perform this work.

Turbulence Modeling and RANS Closures

By Wade Spurlock and Eric Parish, Daniel Israel (Mentor)

Introduction

Simulations of turbulent flow can be computationally expensive. Turbulent fluctuations and energy cascade down to very small length scales, and Direct Numerical Simulation (DNS) is not possible for almost all engineering flows. Instead of resolving all turbulent scales, portions of its energy spectrum are often modeled. Turbulence modeling involves calculating fluctuation correlation terms that arise when some averaging operation is performed on the Navier-Stokes equations. The process of averaging leads to the Reynolds-Averaged Navier-Stokes (RANS) equations, and RANS closure models approximate the physics of the entire turbulent spectrum.

RANS models have varying levels of complexity, from algebraic expressions to transport equations. Additionally, RANS models have been developed to accomidate various types of physical flows. One such set of models are the Besnard-Harlow-Rauenzahn (BHR) models [4]. The BHR models were developed at LANL for variable-density turbulence, such as supernovae and internal confinement fusion (ICF). BHR models use a Favre average (a density-weighted average) to decompose the flow field into mean and fluctuation terms. However, additional density-fluctuation correlation terms appear (opposed to traditional Reynolds averaging) that must be modeled. Different BHR models handle these closure terms with different levels of complexity, but currently the most commonly used BHR models are BHR-2 and BHR-3 [61].

The additional terms and equations required for variable-density turbulence lead to additional model coefficients that must be calibrated. Although RANS simulations are orders of magnitude cheaper than DNS, repeated full-field RANS simulations can be expensive, especially for parametric studies and calibrating model coefficients.

For flows that reach a self-similar state, RANS model coefficients are traditionally calibrated by comparing full-field simulation results to experimental and DNS results. Daniel Israel [30] used an integral method to remove the spatial coordinate from the problem and examine flow quantities in time. This is a much less expensive approach that results in ODEs characterizing the dynamic behavior of the system. These dynamic systems can be plotted directly as flow maps to show trajectories.

This work focuses on validation of the RANS integral method as a faster, cheaper way to compute the transient behavior and steady state of a RANS model. We are comparing RANS model integral method results to xRage simulations. Applying an integral method to higher dimensional RANS models results in a system of three or more ODEs. Consequently, visualizing results is problematic. Simply looking at a single slice of data does not show variation in other variables. For integral method systems of three or more ODEs, we use further mathematical analysis to capture the dominant behavior with variation in all RANS model solution variables. Our approach uses an eigenvalue decomposition to identify fast- and slow-decaying solution components. The two dominant components are visualized along with xRage simulation results postprocessed and projected into the eigenspace.

The following sections describe the analytic and computational methods; then, we present results for model problems: the pure shear (Kelvin-Helmholtz) and pure buoyancy (Rayleigh-Taylor) mixing layers. Each problem is addressed with two different BHR models, with results grouped by integral method, eigenspace decomposition (where applicable), and xRage simulation.

Analytic and Computational Procedures

First, we present a general overview of the analytic approach of the integral method and eigenspace decomposition. Finally, we briefly describe our use of the xRage code.

The Integral Method

The integral method applied to RANS models consists of the following steps:

- Formulate the RANS governing equations and simplify based on the problem
- Integrate governing equations across the layer
- Assume profiles for RANS model variables ($\bar{u}$, k, S, etc).
- Rescale based on layer width $h(t)$, velocity, and time scales

Profiles computed from full-field xRage simulations are presented in the results sections below; generally, density and velocity are assumed to have linear profiles, and turbulent quantities are assumed to follow parabolic profiles.

Eigenspace Decomposition

The general process involves linearizing the nonlinear system of ODEs; then, for the coupled linear ODE system, the coefficient matrix is diagonalized to decouple the system. The eigenvalues of the linear coefficient matrix characterize the system's behavior near the fixed point.

For the general solution vector $\vec{U}$, the nonlinear ODE system from the integral method is written as

$$\frac{d\vec{U}}{dt} = \vec{F}\left(\vec{U}, t; C_{model}\right), \tag{250}$$

where the RHS is written as a general function of the dependent variable $\vec{U}$ and time t, and parameterized by model coefficients C_{model}. Our integral method ODE systems actually do not depend explicitly on time on the RHS, so t is subsequently dropped. The coefficient parameters will not be written for brevity but are included in the equations.

Then, choose some reference state $\vec{U}_{ref}$ and express the solution vector as the sum of the reference vector and some perturbation: $\vec{U} = \vec{U}_{ref} + U'$. The system of ODE's becomes

$$\frac{d\vec{U}_{ref} + U'}{dt} = \vec{F}\left(\vec{U}_{ref} + U'\right). \tag{251}$$

Let the nonlinear RHS function be expanded about $\vec{U}_{ref}$ for some perturbation U' (only retaining the linear term),

$$\vec{F}\left(\vec{U}_{ref} + U'\right) = \vec{F}\left(\vec{U}_{ref}\right) + \left.\frac{\partial \vec{F}}{\partial \vec{U}}\right|_{\vec{U}_{ref}} U'. \tag{252}$$

$\vec{U}_{ref}$ is chosen as a constant reference. In general, since $\frac{d\vec{U}_{ref}}{dt} = 0$, this would lead to an inhomogeneous linear system of ODEs for U',

$$\frac{dU'}{dt} = \vec{F}\left(\vec{U}_{ref}\right) + \left.\frac{\partial \vec{F}}{\partial \vec{U}}\right|_{\vec{U}_{ref}} U'. \tag{253}$$

The ODE system can be further simplified when $\vec{U}_{ref}$ is chosen as an equilibrium solution of the nonlinear RHS function. That is, $\vec{F}\left(\vec{U}_{eq}\right) = 0$, or $\vec{U}_{eq}$ is a root of the nonlinear RHS function. Then, let the Jacobian $\frac{\partial \vec{F}}{\partial \vec{U}}$ be evaluated at the equilibrium solution, resulting in a constant coefficient matrix A,

$$\left.\frac{\partial \vec{F}}{\partial \vec{U}}\right|_{\vec{U}_{eq}} = [A]. \tag{254}$$

The coefficient matrix A will generally be full, leading to a coupled linear system of ODE's for the perturbation U',

$$\frac{dU'}{dt} = [A]\,U'. \tag{255}$$

Going one step further, the coefficient matrix A can be expressed with the diagonal decomposition $A = Y\Lambda Y^{-1}$, where Y is the eigenvector matrix and Λ is the diagonal matrix of eigenvalues. This leads to

$$\frac{dU'}{dt} = [Y]\,[\Lambda]\,[Y]^{-1}\,U'. \tag{256}$$

The solution perturbation vector U' is then transformed as $W' = Y^{-1}U'$, leading to the decoupled linear system of ODE's

$$\frac{dW'}{dt} = [\Lambda]\,W'. \tag{257}$$

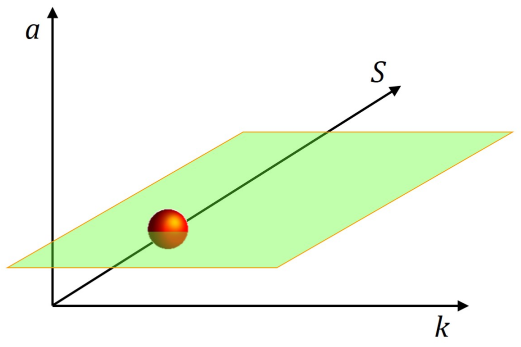
The solution of the decoupled system determines how each coefficient of the eigenvector expansion evolves in time. That is,

$$U'(t) = YW'(t) = w_1(t)\vec{y}_1 + w_2(t)\vec{y}_2 + \ldots + w_N(t)\vec{y}_N, \tag{258}$$

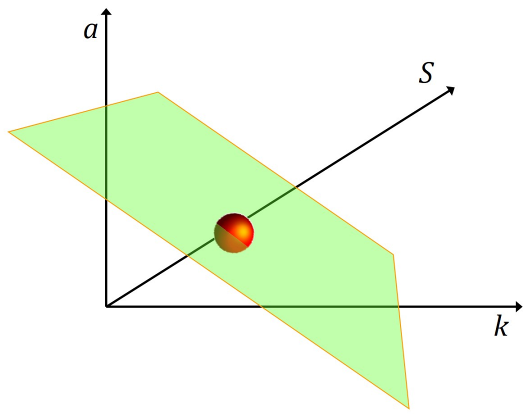
and the solution of the decoupled system is $w_i(t) = w_i(0)\exp(\lambda_i t)$, for each component $1 \leq i \leq N$. The eigenvalues λ_i are generally complex-valued, with the real component determining exponential decay ($Re(\lambda_i) < 0$) or growth ($Re(\lambda_i) > 0$), and the imaginary component determining oscillation (with frequency $Im(\lambda_i)$).

With the physical solution perturbation U' projected into the basis spanned by eigenvectors Y, we can identify coefficients $w_i(t)$ that decay more quickly than others and, hopefully, consider a smaller dimensional space truncated at k components, $\widetilde{w}_j(t)$, where $1 \leq j \leq k \leq N$. For visualizing the system in two dimensions, the eigenspace is truncated to the two slow-decaying components.

A slightly more abstract concept, at least in four or more dimensions, is what we are calling the "truncated eigenspace reconstruction plane" - it is the plane spanned by eigenvectors corresponding to slow-decaying eigenvalues. In physical space, this is the plane that trajectories approach as they go to the fixed point. In Figure 199, we show sketches to illustrate this concept. In a simple $k-S$ slice at a constant value of a, we would lose all out-of-plane information and variation in a. However, viewing the physical plane reconstructed from the truncated eigenspace, we are capturing variation in all physical variables. By projecting trajectories onto this plane, we would expect to capture more accurate results and can compare the full nonlinear trajectories to those in the truncated plane.



(a) $k-S$ plane cut, a fixed



(b) truncated eigenbasis reconstruction plane

Figure 199: Sketches of different planes on which to view trajectories

Finally, if eigenvectors are complex conjugates, eigencoefficients are complex conjugates and truncated solution is real-valued. Since the Jacobian matrix is real-valued, if an eigenvalue is complex it must have a conjugate pair. Of course, the corresponding eigenvectors would appear in complex conjugate pairs as well. If we then have an N-dimensional problem in real space, the eigenvalue decomposition transforms the problem into N-dimensional complex space. However, since complex-valued eigenvalues must occur in conjugate pairs, we are actually not increasing the number of parameters. Furthermore, if the slow-decaying eigencoefficients are a complex conjugate pair, we can simply plot one of them in the complex plane since only two parameters define both of the coefficients. Then, since the corresponding eigenvectors are also complex conjugates, the reconstructed physical solution will be real-valued.

xRage

We are using xRage, an Eulerian code developed at LANL. xRage has many features, such as adaptive mesh refinement, plasma models, material strength, and radiation transport, just to name a few. Our simulations utilize the turbulence mix models in xRage, which are various versions of Besnard-Harlow-Rauenzahn (BHR) models for variable-density turbulence.

Pure Shear Layer $(k-S)$

Integral Equations

For the temporal shear case the governing equations reduce to [31],

$$\frac{\partial \bar{u}}{\partial t} = \frac{\partial}{\partial y}\left[\nu_T \frac{\partial \bar{u}}{\partial y}\right], \tag{259a}$$

$$\frac{\partial k}{\partial t} = -R_{xy}\frac{\partial \bar{u}}{\partial y} + \frac{\partial}{\partial y}\left[\frac{\nu_T}{\mathrm{Pr}_T}\frac{\partial k}{\partial y}\right] - \frac{k^{3/2}}{S}, \tag{259b}$$

$$\frac{\partial S}{\partial t} = \frac{-S}{k}\left(\frac{3}{2} - C_{\varepsilon 1}\right) R_{xy}\frac{\partial u}{\partial y} - \left(\frac{3}{2} - C_{\varepsilon 2}\right)\sqrt{k} + \frac{\partial}{\partial y}\left[\frac{\nu_T}{\mathrm{Pr}_S}\frac{\partial S}{\partial y}\right] \tag{259c}$$

The following profiles are assumed,

$$\bar{u} = \frac{\Delta U}{2}\eta \qquad k = \hat{k}(t)(1-\eta^2) \qquad S = \hat{S}(t)\sqrt{1-\eta^2}$$

where $\eta = \frac{y}{h}$ is the similarity variable. We non-dimensionalize by the following,

$$k^* = \frac{\hat{k}}{\Delta U^2} \qquad S^* = \frac{\hat{S}}{h} \qquad \frac{dt^*}{dt} = \frac{\Delta U}{h}$$

And we obtain:

$$\frac{dh}{dt} = 2C_\mu \sqrt{k^*}\Delta U S^* \tag{260a}$$

$$\frac{dk^*}{dt^*} = C_\mu\left(\frac{1}{4} - 2k^*\right)\sqrt{k^*}S^* - \frac{k^{*3/2}}{S^*} \tag{260b}$$

$$\frac{dS^*}{dt^*} = C_\mu\left(\frac{3}{8} - 4k^* - \frac{C_{\varepsilon 1}}{4}\right)\frac{S^{*2}}{\sqrt{k^*}} + \sqrt{k^*}\left(C_{\varepsilon 2} - \frac{3}{2}\right) \tag{260c}$$

The k and S equations have now become decoupled from the h equation, so they form a closed set. The fixed point of the set is found by setting the LHS to zero and is,

$$k^* = \frac{C_{\varepsilon 2} - C_{\varepsilon 1}}{8\,C_{\varepsilon 2} + 4}$$

$$S^* = \sqrt{\frac{C_{\varepsilon 2} - C_{\varepsilon 1}}{2\,C_\mu\,C_{\varepsilon 1} + C_\mu}}$$

With the following coefficients,

$$C_\mu = 0.09, C_{\varepsilon 1} = 1.44, C_{\varepsilon 2} = 1.92$$

the fixed point is,

$$k^* = 0.025$$

$$S^* = 1.17$$

We can also look at (260) as a dynamical system, and plot the trajectories on a flow map, as shown in Figure 200. The dot is the fixed point, which we can see attracts over the applicable range for the $k - S$ model. Note that the fixed point corresponds to the self-similar solution of the original PDEs.

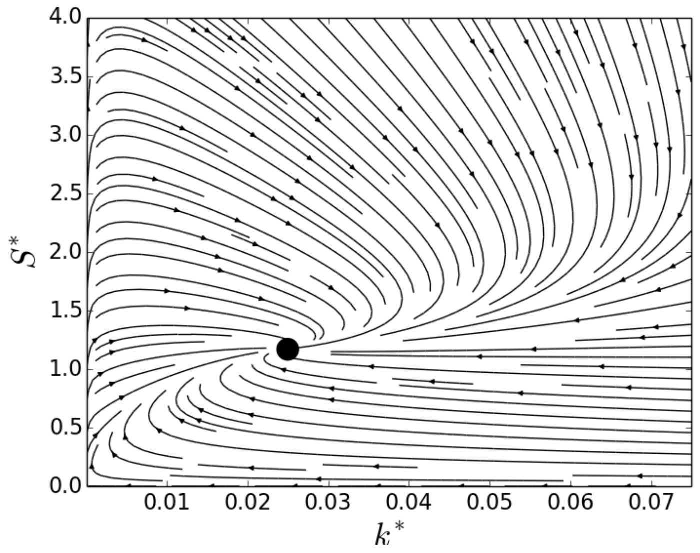


Figure 200: Flow for the systems in Eq 260.

Eigenspace Analysis

Since a two-dimensional decoupled system is obtained through the integral method there is no need to reduce the parameter space of the model. It is, however, still of interest to observe the character of the fixed point. We take the solution vector to be,

$$\vec{U} = \begin{pmatrix} k^* \\ S^* \end{pmatrix}$$

The Jacobian evaluated at the fixed point is,

$$A = \begin{bmatrix} -0.1675 & 0.0057 \\ -0.4753 & -0.1128 \end{bmatrix}$$

which gives complex conjugate eigenvalues,

$$\lambda_{1,2} = -0.044 \pm 0.140i$$

Since the real portion of the eigenvalue is negative the solution approaches a fixed point, which is as one would expect. However the imaginary portion of the eigenvalues causes the solution to spiral infinitely into the fixed point, which is not physically obvious. This could simply be a characteristic of the $k-S$ turbulence model, or suggestive of actual physics.

xRage Comparison

To test the integral method presented above, a temporal shear case was simulated in xRage. This case was run for the various initializations of k and S seen in Table 8. Default xRage model coefficients for BHR-2 were used except for those presented in Table 8. We note that for initialization S_0 can not be non-dimensionalized by the layer width since it is infinitely thin. Instead, following Stalsberg-Zarling and Gore [66], $S_0^* = S_0/0.1dx$. The non-dimensionalization of k_0^* remains $k_0^* = k_0/\Delta u^2$. A grid spacing of $\Delta x = 0.0078125$cm was used.

	Run 1	Run 2	Run 3	Run 4	Run 5
k_0^*	0.01	0.005	0.01	0.01	0.1
S_0^*	0.001	100	500	1000	1000
C_μ	0.09	0.09	0.09	0.09	0.09
Pr_S	1.0	1.0	1.0	1.0	1.0
Color (Fig. 201a)	Blue	Green	Red	Purple	Teal

Table 8: Summary of initializations for X-Rage runs.

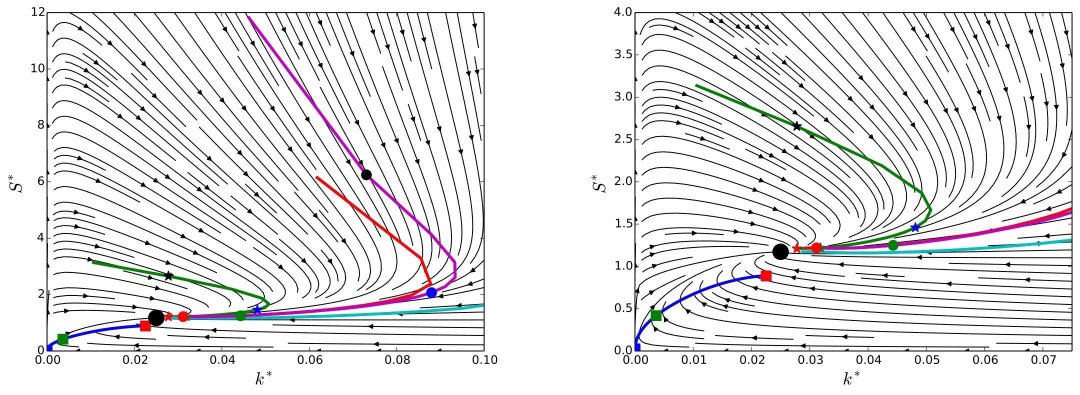
The trajectories of the RANS runs are overlayed on the flow of the dynamical systems in Figures 201a and 201b. The layer half-width is found from the momentum thickness.

$$\theta = \int_{-\infty}^{\infty} (u-u_1)(u_2-u)dy$$

By inserting the assumed velocity profile we find the relation,

$$h = 3\theta$$

Note that in Figures 201a and 201b some of the early time steps have been cut off of to avoid transients from the initial conditions. Additionally, for runs initialized with a high initial length scale, the the timescale at which S approaches self-similarity is very long. The turbulent kinetic energy profile reaches self-similarity faster, leaving little turbulent kinetic energy outside of the layer. Due to this the S profile decays very slowly, causing the runs that were initialized with a high initial length scale to not be fully self-similar at the end of the run.



(a) Larger domain in $k^* - S^*$ (b) Zoomed in around the fixed point

Figure 201: Flow for the dynamical systems. The thick lines are trajectories for X-Rage simulations with a variety of initializations for k^* and S^*. The colors of the markers correspond to the colors of the full profiles plotted below.

Overall good agreement is seen between the integral approximation and the RANS simulations. Away from the fixed point the RANS trajectories differ slightly from their integral approximations. As the profiles continue to develop, the trajectories begin to coincide and approach the fixed point. We do notice that Run 1 (blue in Figures 201a and 201b) appears to approach a different solution and miss the fixed point. In fact, it appears that the profile for S converges to a different solution, which is seen in Figure 202. The length scale profiles for higher initializations of S seem to "fall" upon the assumed profile as time progresses. The same profiles for low initializations of S follow a considerably different pattern, as we see in Figure 202. The exact reason for this is not known. It could be that once the runs with high initial length scales become fully self-similar the two profiles will coincide, but this is not apparent in Figure 202.

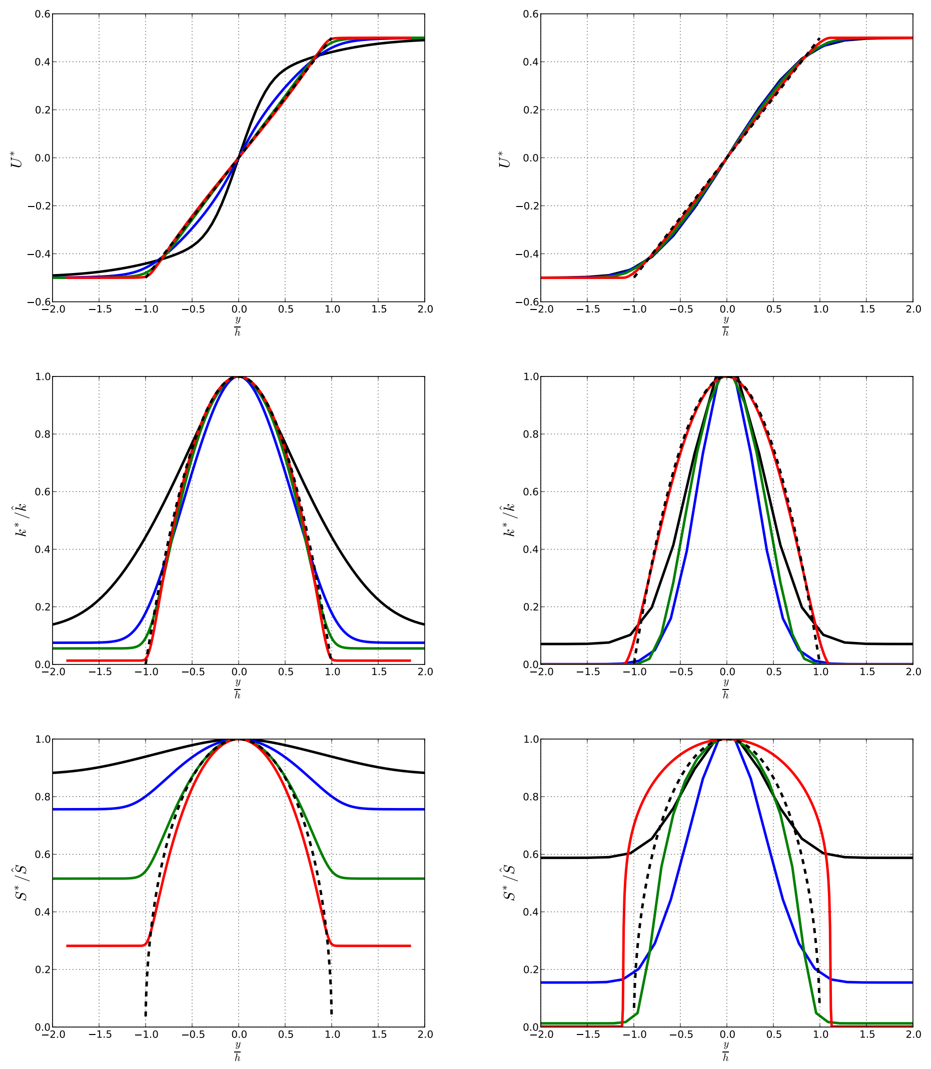


Figure 202: The profiles develop differently for initializations of $S = 1000.0$ (left) and $S = 0.001$ (right). The dashed line is the guessed profile.

In Figure 202 we can see there is excellent agreement between the guessed and calculated profiles for U and k. For the high initialization of S good agreement is also observed for the S profile. Since these profiles match well there is no need to add additionally parameters to the guessed profiles in the integral method. However, for different model coefficients and different types of flows, different profiles may need to be considered. This is seen in Figure 203, where we observe how different coefficients change the shape of the profiles.

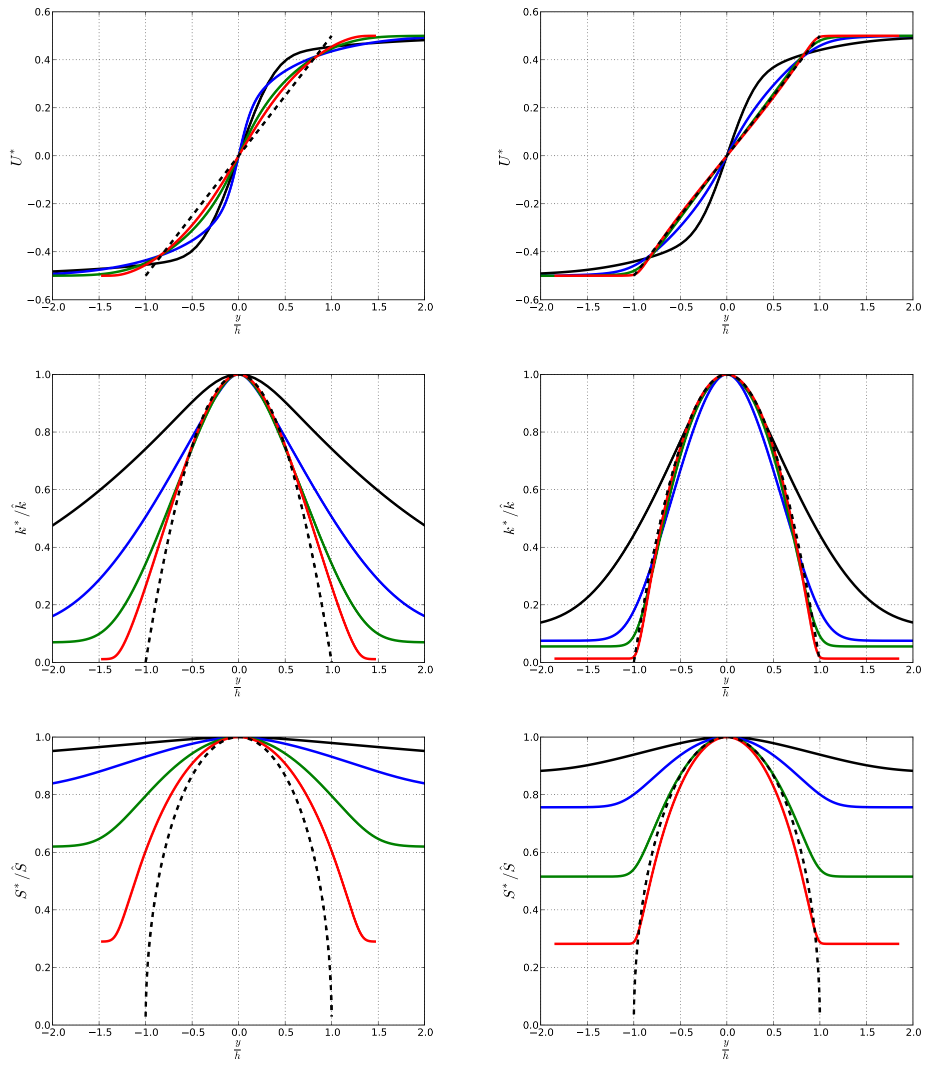


Figure 203: The profiles develop differently for $C_\mu = 0.28$ (left) and $C_\mu = 0.09$ (right). When $C_\mu = 0.09$ the profiles are less diffusive and adhere more to the assumed profiles.

Pure Shear Layer (BHR3)

Integral Equations

For a single fluid without density variation and undergoing plane shear, BHR-3 reduces to the full Reynolds stress model equations [31]:

$$\frac{\partial \overline{u}}{\partial t} = -\frac{\partial R_{xy}}{\partial y} \tag{261a}$$

$$\frac{\partial R_{yy}}{\partial t} = \frac{\partial}{\partial y}\left[C_r \frac{S}{\sqrt{k}} R_{yy} \frac{\partial R_{yy}}{\partial y}\right] - C_{r3}\frac{\sqrt{k}}{S}\left(R_{yy} - \frac{2}{3}k\right) - C_{r2}\frac{2}{3}R_{xy}\frac{\partial \overline{u}}{\partial y} - \frac{2}{3}\frac{k^{3/2}}{S} \tag{261b}$$

$$\frac{\partial R_{xy}}{\partial t} = \frac{\partial}{\partial y}\left[C_r \frac{S}{\sqrt{k}} R_{yy} \frac{\partial R_{xy}}{\partial y}\right] + (C_{r2}-1)R_{yy}\frac{\partial \overline{u}}{\partial y} - C_{r3}\frac{\sqrt{k}}{S}R_{xy} \tag{261c}$$

$$\frac{\partial k}{\partial t} = \frac{\partial}{\partial y}\left[C_r \frac{S}{\sqrt{k}} R_{yy} \frac{\partial k}{\partial y}\right] - R_{xy}\frac{\partial \overline{u}}{\partial y} - \frac{k^{3/2}}{S} \tag{261d}$$

$$\frac{\partial S}{\partial t} = \frac{\partial}{\partial y}\left[C_S \frac{S}{\sqrt{k}} R_{yy} \frac{\partial S}{\partial y}\right] - \frac{S}{k}\left(\frac{3}{2} - C_1\right)R_{xy}\frac{\partial \overline{u}}{\partial y} - \sqrt{k}\left(\frac{3}{2} - C_2\right) \tag{261e}$$

The following profiles are assumed,

$$\bar{u} = \frac{\Delta U}{2}\eta \qquad k = \hat{k}(t)(1-\eta^2) \qquad S = \hat{S}(t)\sqrt{1-\eta^2}$$

$$R_{yy} = \hat{R}_{yy}(t)(1-\eta^2) \qquad R_{xy} = \hat{R}_{xy}(t)(1-\eta^2)$$

where $\eta = \frac{y}{h}$ is the similarity variable. We non-dimensionalize by the following,

$$k^* = \frac{\hat{k}}{\Delta U^2} \qquad S^* = \frac{\hat{S}}{h} \qquad R^*_{yy} = \frac{\hat{R}_{yy}}{\Delta U^2} \qquad R^*_{xy} = \frac{\hat{R}_{xy}}{\Delta U^2} \qquad \frac{dt^*}{dt} = \frac{\Delta U}{h}$$

and the integral method system of ODEs is

$$\frac{1}{\Delta U}\frac{dh}{dt} = -4R^*_{xy} \tag{262a}$$

$$\frac{dR^*_{yy}}{dt^*} = R^*_{xy}\left(4R^*_{yy} - \frac{1}{3}C_{r2}\right) + \frac{k^{*1/2}}{S^*}\left(\frac{2}{3}(C_{r3}-1)k^* - C_{r3}R^*_{yy}\right) \tag{262b}$$

$$\frac{dR^*_{xy}}{dt^*} = \frac{1}{2}(C_{r2}-1)R^*_{yy} + 4R^{*2}_{xy} - C_{r3}\frac{k^{*1/2}R^*_{xy}}{S^*} \tag{262c}$$

$$\frac{dk^*}{dt^*} = \left(4k^* - \frac{1}{2}\right)R^*_{xy} - \frac{k^{*3/2}}{S^*} \tag{262d}$$

$$\frac{dS^*}{dt^*} = \left(\frac{C_1}{2} + 8k^* - \frac{3}{4}\right)\frac{R^*_{xy}S^*}{k^*} + \left(C_2 - \frac{3}{2}\right)k^{*1/2} \tag{262e}$$

With the following coefficients,

$$C_1 = 1.2, C_2 = 1.77, C_{r2} = 0.6, C_{r3} = 1.8$$

the fixed point is,

$$R^*_{yy} = 0.0157$$

$$R^*_{xy} = -0.0111$$

$$k^* = 0.0314$$

$$S^* = 1.338$$

Eigenspace Analysis

For the solution vector

$$\vec{U} = \begin{pmatrix} R^*_{yy} \\ R^*_{xy} \\ k^* \\ S^* \end{pmatrix},$$

$$\frac{d\vec{U}}{dt} = \vec{F}\left(\vec{U}, t; C_1, C_2, C_{r2}, C_{r3}\right), \tag{263}$$

The Jacobian evaluated at the fixed point is,

$$A = \begin{bmatrix} -0.2827 & -0.1372 & 0.0464 & 0.0011 \\ -0.2000 & -0.3271 & 0.0421 & -0.0020 \\ 0 & -0.3744 & -0.2430 & 0.0031 \\ 0 & 4.3097 & -1.4991 & -0.0358 \end{bmatrix}$$

which gives complex conjugate eigenvalues,

$$\lambda_{1,2} = -0.3626 \pm 0.0856i, \lambda_{3,4} = -0.0817 \pm 0.0932i$$

The corresponding eigenvectors are,

$$Y = \begin{bmatrix} -0.075 - 0.055i & -0.075 + 0.055i & -0.005 + 0.0.017i & -0.005 - 0.0.017i \\ -0.083 - 0.102i & -0.083 + 0.102i & 0.006 - 0.010i & -0.006 + 0.010i \\ -0.037 - 0.345i & -0.037 + 0.345i & -0.014 + 0.032i & -0.014 - 0.032i \\ 0.924 & 0.924 & -0.999 & -0.999 \end{bmatrix}$$

Due to the seperation in the eigenvalues, the system will be composed of two fast-decaying eigencoefficients and two slow-decaying eigen-coefficients. Instead of visualizing the system 4D system by a basic cross-sectional cut-out, we can reconstruct the plane on which the fast decaying eigencoefficients are zero, as described earlier. Flow maps for this reconstruction are shown in Figure 204.

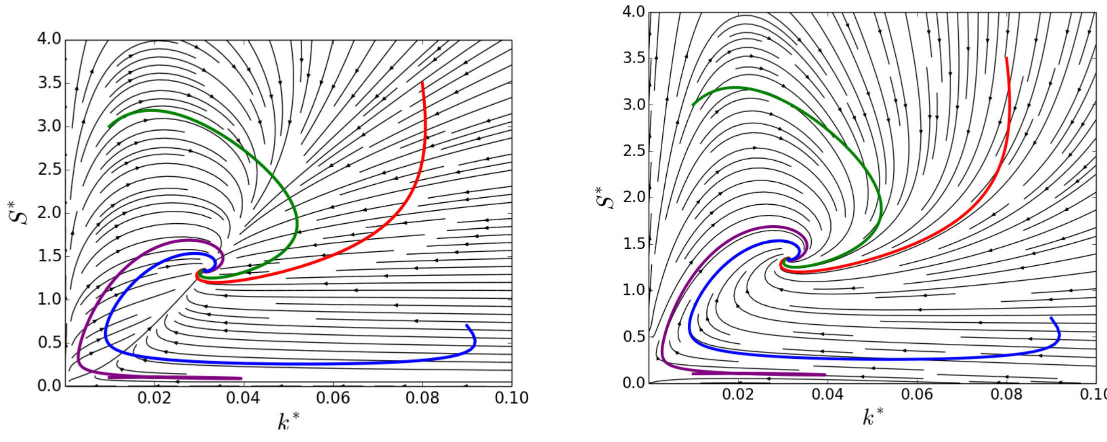


Figure 204: Analytic solutions shown at a cross-section where R_{yy} and R_{xy} are at their fixed point (left) and on the plane of the dominant eigencoefficients (right). The solution on the plane of the dominant eigencoefficients is a more accurate representation of the solution since it falls onto this plane.

We can also construct a trajectory plot of the dominant eigencoefficients. For this BHR-3 problem, the dominant eigencoefficients are a complex conjugate pair, so we simply plot one of them in the complex plane. Figure 205 shows four subfigures, one for each physical space quantity, with contours of relative distance to the fixed point plotted over the same eigencoefficient trajectories. We can gain physical insight even from the mathematically transformed system. Two key points are which region in eigenspace corresponds to physically allowable variables, and which variables change similarly or differently along the trajectories.

First, we have already assumed that k, S, and R_{yy} are positive and R_{xy} is negative. However, the relative distance or each physical variable is normalized by the fixed point value - that is, $d_{relative} = \frac{f - f_{fixed-point}}{f_{fixed-point}}$, where f represents the physical space variables. With this formulation, when the relative distance is -1, the physical variable is equal to zero. What is even more important is that when $d_{relative} < -1$, the physical space variable has the opposite sign of the fixed-point value, which means that it is not physically allowable. In Figure 205 a bold magenta line is along $d_{relative} < -1$ for each variable, and the unphysical region is denoted by the large red X. We do see unusual trajectories in these regions that actually move away from the fixed point or seem to approach singularities; however, these are not physical, should not be considered, and would not be encountered in actual simulations.

Second, we can see which variables change in a similar fashion along the dominant eigenvector trajectories. Specifically, as seen in Figure 205, the contours of k, R_{yy}, and R_{xy} are all nearly aligned, while the contours S are exactly vertical. These angles can be worked out from the eigenvectors corresponding to the dominant eigenvalues, and we immediately see that the fourth component of the eigenvectors, related to S, is real-valued. Therefore, variations in the imaginary component of the eigencoefficient have no effect on the value of S. In a mathematical sense, for the j^{th} component of the solution vector $\vec{U} = \left[R^*_{yy}\ R^*_{xy}\ k^*\ S^*\right]^T$, the angle in the eigencoefficient plane is determined by the real and imaginary components of the corresponding eigenvector, or $\theta_j = \arctan\left(\frac{Re(y_{dom,j})}{Im(y_{dom,j})}\right)$. The interesting observation is that k, R_{yy}, and R_{xy}

all vary in a similar fashion along the trajectories (the angles are not exactly equal, but within $\sim 5°$). This makes physical sense, as these quantities are all related to velocity fluctuations while S is a different physical process.

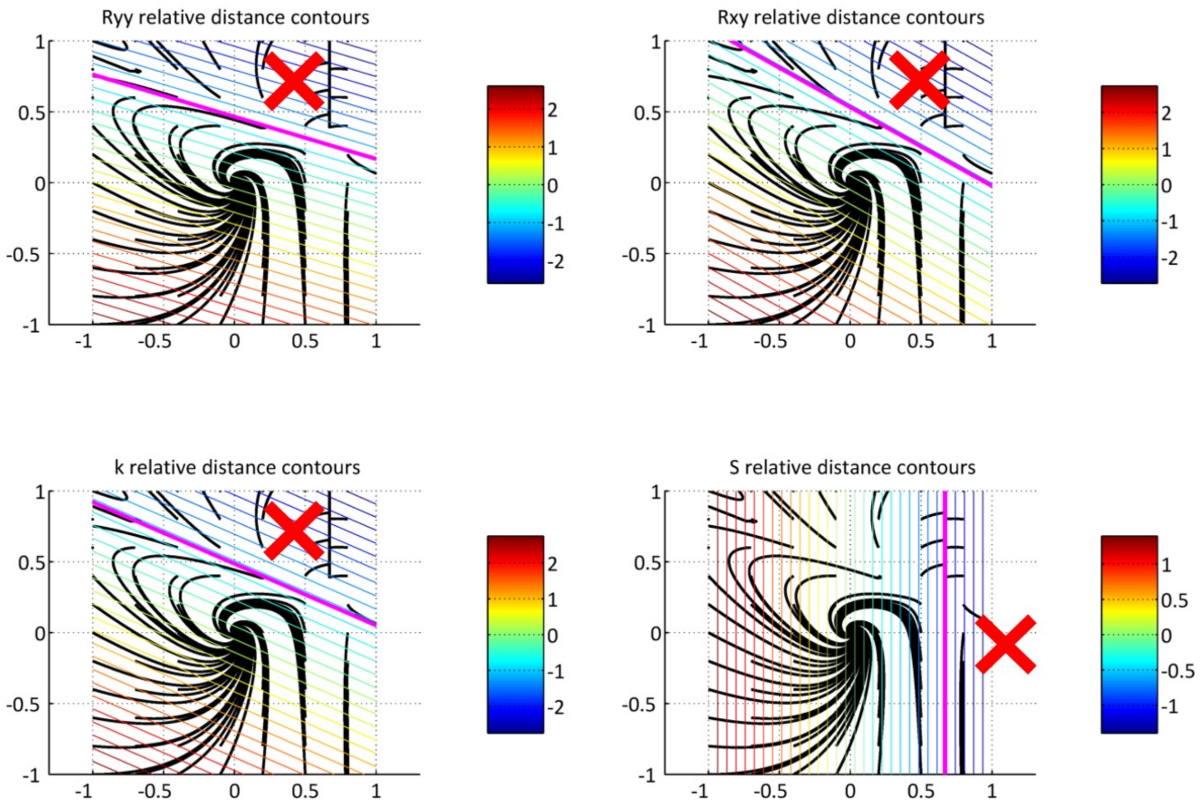


Figure 205: Dominant eigencoefficient trajectories with contours of normalized distance to fixed point for each quantity in physical space.

xRage Comparison

Default xRage coefficients were used except for those shown in Table 9. The same grid spacing and domain that was used in the BHR-2 case was used here.

	Run 1	Run 2	Run 3
k_0^*	0.01	0.01	5.0
S_0^*	0.001	1000	500
C_μ	0.09	0.09	0.09
Pr_S	1.0	1.0	1.0
Color (Fig. 207)	Blue	Red	Green

Table 9: Summary of initializations for X-Rage runs.

Slow numerical convergence was observed for the xRage simulations that used the BHR-3 turbulence model. This is caused by the time-stepping in xRage to switch to a BHR time-step, which was several orders of magnitude lower than the default CFL time-step. Runs with a forced time-step at the starting CFL time-step went unstable at late time. Due to this the runs initialized with higher initial length scales were not ran long enough to reach self-similarity.
In order to visualize the solution, the eigenspace decomposition described in the above section was used. We can visualize most of the system by considering a cross-section in eigenspace on which the fast decaying eigencoefficients are zero.

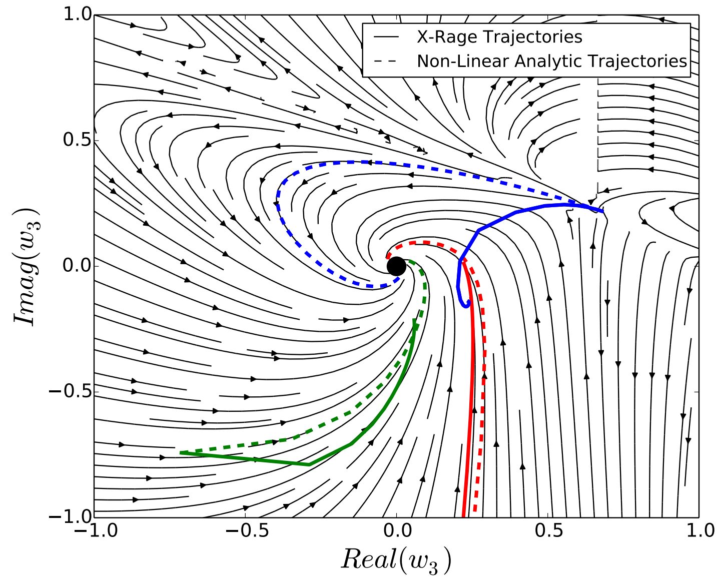


Figure 206: xRage trajectories projected onto a cross-section in eigenspace on which the fast decaying eigencoefficients are zero.

In Figure 206 we see good agreement between xRage and the analytic flow map, with the exception of Run 1 (blue). The flow map accurately describes the trajectories from early time, since little out of plane information is lost. A comparison of the runs in physical space on the plane in which the fast decaying eigencoefficients are zero is shown in Figure 207.

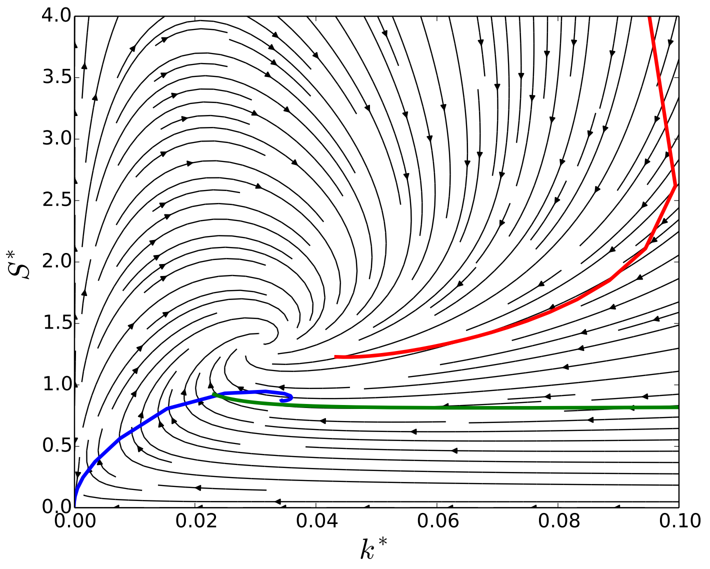


Figure 207: The $k-S$ components of xRage trajectories projected onto the plane of the dominant eigencoefficients in physical space.

Again, good agreement is seen between the xRage and analytic trajectories. The xRage profiles for runs initialized with high length-scales don't quite reach the fixed point (due to the fact that they were not run long enough because of the slow convergence), but the trajectories approach the fixed point as predicted. For the run with a low initialization of S, the solution misses the fixed point in the same manner as it did in the BHR-2 runs. Once again this is caused by the profiles converging to a shape different than what was used in the integral method. However the same qualitative behavior is observed.

Rayleigh Taylor Layer ($k-S-a$)

Integral Equations

The governing equations with a $k-S-a$ turbulence model are [31],

$$\frac{\partial \bar{\rho}}{\partial t} = \frac{\partial}{\partial y}\Big[\nu_T \frac{\partial \bar{\rho}}{\partial y}\Big], \tag{264a}$$

$$\frac{\partial k}{\partial t} = \frac{a_y}{\rho}\frac{\partial \bar{\rho}}{\partial y} + \frac{\partial}{\partial y}\Big[\frac{\nu_T}{\mathrm{Pr}_T}\frac{\partial k}{\partial y}\Big] - \frac{k^{3/2}}{S}, \tag{264b}$$

$$\frac{\partial S}{\partial t} = \frac{S}{k}\Big(\frac{3}{2} - C_3\Big)a_y\frac{\partial \bar{\rho}}{y} + \frac{\partial}{\partial y}\Big[\frac{\nu_T}{\mathrm{Pr}_S}\frac{\partial S}{\partial y}\Big] - \Big(\frac{3}{2} - C_2\Big)\sqrt{k}, \tag{264c}$$

$$\frac{\partial a_y}{\partial t} = \frac{b}{\rho}\frac{\partial \bar{\rho}}{\partial y} - R_{yy}\frac{\partial \bar{\rho}}{\partial y} + \frac{3}{2}\frac{\partial a_y^2}{\partial y} + C_a\frac{\partial}{\partial y}\Big[\frac{\partial \nu_T}{\mathrm{Pr}_S}\frac{\partial a_y}{\partial y}\Big] - C_{a1}\frac{\sqrt{k}}{S}a_y \tag{264d}$$

The following profiles are assumed,

$$\bar{\rho} = \rho_0 + \frac{\Delta\rho}{2}\eta \qquad k = \hat{k}(t)(1-\eta^2) \qquad S = \hat{S}(t)\sqrt{1-\eta^2}$$

$$a_y = \hat{a}_y(t)(1-\eta^2) \qquad b = \hat{b}(t)(1-\eta^2)$$

and non-dimensionalized by,

$$k^* = \frac{\hat{k}}{\mathrm{At}gh} \qquad S^* = \frac{\hat{S}}{h} \qquad a_y^* = \frac{\hat{a}_y}{\mathrm{At}^{3/2}\sqrt{gh}}$$

$$b^* = \frac{\hat{b}}{\mathrm{At}^2} \qquad \frac{dt^*}{dt} = \sqrt{\frac{\mathrm{At}g}{h}}$$

We now integrate across the layer (this time taking the first moment of the continuity equation) and non-dimensionalize to get,

$$\frac{\partial k^*}{\partial t} = -k^{3/2}\left(\frac{1}{S^*} + 4C_\mu \frac{S^*}{\mathrm{Pr}_T}\right) - a_y^*, \tag{265a}$$

$$\frac{\partial S^*}{\partial t} = \sqrt{k^*}\left(C_2 - \frac{3}{2} - 4\frac{C_\mu}{\mathrm{Pr}_T}S^{*2}\right) + \left(\frac{3}{2} - C_3\right)\frac{S^*}{k^*}a_y^*, \tag{265b}$$

$$\frac{\partial a_y^*}{\partial t} = -\sqrt{k^*}\left(3\frac{C_\mu}{\mathrm{Pr}_T}S^* a_y^* + C_{a1}\frac{a_y^*}{S^*}\right) - \frac{2}{3}k^* - b^* \tag{265c}$$

Although this is a three-dimensional system and can be visualized in physical space, 3D plots do not offer the same insight into the system and are considerably more confusing. Since it is difficult to visualize, we will consider an eigenspace analysis to see if we can reduce the dimensionality of the system.

Eigenspace Analysis

From existing data, we assume $b^* = 0.25$. The fixed point is found numerically to be,

$$k^* = 0.1555 \qquad S^* = 0.7423 \qquad a_y^* = -0.1676$$

We take our solution vector to be,

$$\vec{U} = \begin{pmatrix} k^* \\ S^* \\ a_y^* \end{pmatrix}$$

The Jacobian evaluated at the fixed point is,

$$A = \begin{bmatrix} -1.617 & -0.003 & -1 \\ -2.315 & -0.770 & -1.432 \\ 0.470 & -0.291 & -2.110 \end{bmatrix}$$

The eigen-value matrix is,

$$\lambda = \begin{bmatrix} -1.09+0.316i & 0 & 0 \\ 0 & -1.09-0.316i & 0 \\ 0 & 0 & -2.328 \end{bmatrix}$$

There is little separation between the eigenvalues, meaning that truncating the system will lose considerable information.

xRage Comparison

Rayleigh-Taylor simulations were ran in xRage. The following parameters were used for every case,

dx	Domain	C_a	C_{a2}	C_C	C_{a1}	C_{r1}	C_{r2}	C_{r3}	C_{r4}	C_{b2}	C_4	C_S	C_1	C_3
0.03125 cm	40 cm	0.3	0.25	0.56	3.2	0.3	0.6	0.42	1.8	2.0	1.2	4.2	1.2	0.0

Table 10: Summary of coefficients for xRage Rayleigh-Taylor runs.

Runs were initialized by varying k and S to be,

	Run 1	Run 2	Run 3
k_0^*	0.25	1e-5	1e-6
S_0^*	0.010	1e-3	1e-4

Table 11: Summary of initializations for xRage Rayleigh-Taylor runs.

Since this is a 3D system, we can visualize it in physical space. Figure 208 shows a comparison of analytic and xRage trajectories.

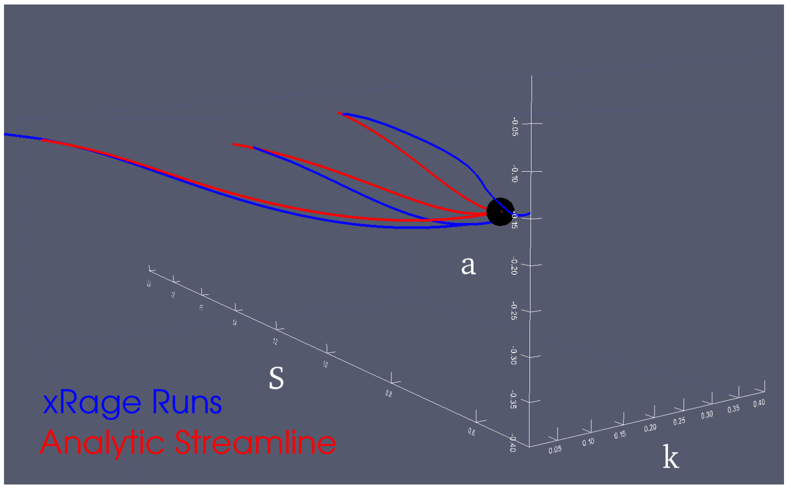


Figure 208: A three-dimensional visualization comparing analytic and xRage trajectories.

Although Figure 208 can be used to qualitatively compare the two trajectories, it is still difficult to analyze their accuracy. Due to the lack of separation of the eigenvalues, the system can't be easily truncated either.

Rayleigh Taylor Layer ($k-S-a-b$)

Integral Equations

The governing equations with a $k-S-a-b$ turbulence model are [31],

$$\frac{\partial\bar{\rho}}{\partial t}=\frac{\partial}{\partial y}\big[\nu_T\frac{\partial\bar{\rho}}{\partial y}\big], \tag{266a}$$

$$\frac{\partial k}{\partial t}=\frac{a_y}{\rho}\frac{\partial\bar{p}}{\partial y}+\frac{\partial}{\partial y}\big[\frac{\nu_T}{\mathrm{Pr}_T}\frac{\partial k}{\partial y}\big]-\frac{k^{3/2}}{S}, \tag{266b}$$

$$\frac{\partial S}{\partial t}=\frac{S}{k}(\frac{3}{2}-C_3)a_y\frac{\partial\bar{p}}{y}+\frac{\partial}{\partial y}\big[\frac{\nu_T}{\mathrm{Pr}_S}\frac{\partial S}{\partial y}\big]-(\frac{3}{2}-C_2)\sqrt{k}, \tag{266c}$$

$$\frac{\partial a_y}{\partial t}=\frac{b}{\rho}\frac{\partial\bar{p}}{\partial y}-R_{yy}\frac{\partial\bar{\rho}}{\partial y}+\frac{3}{2}\frac{\partial a_y^2}{\partial y}+C_a\frac{\partial}{\partial y}\big[\frac{\partial\nu_T}{\mathrm{Pr}_S}\frac{\partial a_y}{\partial y}\big]-C_{a1}\frac{\sqrt{k}}{S}a_y, \tag{266d}$$

$$\frac{\partial b}{\partial t}=-2(b+1)a_y\frac{\partial\bar{\rho}}{\partial y}+2a_y\frac{\partial b}{\partial y}+\bar{\rho}C_b\frac{\partial}{\partial y}\big[\frac{\nu_T}{\mathrm{Pr}_S}\frac{\partial b}{\partial y}\big]-C_{b1}\frac{\sqrt{k}}{S}b \tag{266e}$$

The following profiles are assumed,

$$\bar{\rho}=\rho_0+\frac{\Delta\rho}{2}\eta \qquad k=\hat{k}(t)(1-\eta^2) \qquad S=\hat{S}(t)\sqrt{1-\eta^2}$$

$$a_y=\hat{a_y}(t)(1-\eta^2) \qquad b=\hat{b}(t)(1-\eta^2)$$

and non-dimensionalized by,

$$k^*=\frac{\hat{k}}{\mathrm{At}gh} \qquad S^*=\frac{\hat{S}}{h} \qquad a_y^*=\frac{\hat{a_y}}{\mathrm{At}^{3/2}\sqrt{gh}}$$

$$b^*=\frac{\hat{b}}{\mathrm{At}^2} \qquad \frac{dt^*}{dt}=\sqrt{\frac{\mathrm{At}g}{h}}$$

We recall for BHR-2,

$$R_{ij}=-\nu_T\big(\frac{\partial\bar{u}_i}{\partial x_j}+\frac{\partial\bar{u}_j}{\partial x_i}-\frac{2}{3}\delta_{ij}(\nabla\cdot\bar{u})\big)+\frac{2}{3}\delta_{ij}k$$

So $R_{yy}=\frac{2}{3}k$. We now integrate across the layer (this time taking the first moment of the continuity equation) and non-dimensionalize to get,

$$\frac{\partial k^*}{\partial t}=-k^{3/2}(\frac{1}{S^*}+4C_\mu\frac{S^*}{\mathrm{Pr}_T})-a_y^*, \tag{267a}$$

$$\frac{\partial S^*}{\partial t}=\sqrt{k^*}(C_2-\frac{3}{2}-4\frac{C_\mu}{\mathrm{Pr}_T}S^{*2})+(\frac{3}{2}-C_3)\frac{S^*}{k^*}a_y^*, \tag{267b}$$

$$\frac{\partial a_y^*}{\partial t}=-\sqrt{k^*}(3\frac{C_\mu}{\mathrm{Pr}_T}S^*a_y^*+C_{a1}\frac{a_y^*}{S^*})-\frac{2}{3}k^*-b^*, \tag{267c}$$

$$\frac{\partial b^*}{\partial t}=-\sqrt{k^*}(2\frac{C_\mu}{\mathrm{Pr}_T}S^*b^*+C_{b1}\frac{b^*}{S^*})-a_y^*(\frac{8}{5}\mathrm{At}^2b^*+2) \tag{267d}$$

We note that this is a four-component dynamical system. Much like the BHR-3 model in the temporal shear case, there is no easy way to visualize the solution to the system and an eigenspace analysis must be used.

Eigenspace Analysis

We follow the same procedure of linearizing around the fixed point. The following coefficients are used,

$$C_\mu = 0.28 \qquad C_{b1} = 2.0 \qquad \mathrm{Pr}_T = 0.6 \qquad C_{a1} = 3.2 \qquad C_2 = 1.92 \qquad C_3 = 1.2$$

An analytic solution for the fixed point is not trivial, but can easily be found numerically to be,

$$k^* = 0.156 \qquad S^* = 0.744 \qquad {a_y}^* = -0.169 \qquad b^* = 0.253$$

We take our solution vector to be,

$$\vec{U} = \begin{pmatrix} k^* \\ S^* \\ a_y^* \\ b^* \end{pmatrix}$$

The Jacobian evaluated at the fixed point is,

$$A = \begin{bmatrix} -1.622 & -0.004 & -1 & 0 \\ -2.315 & -0.774 & -1.425 & 0 \\ 1.143 & -0.294 & -2.114 & -1 \\ -1.083 & 0.268 & -2.001 & -1.338 \end{bmatrix}$$

The eigenvalue matrix is,

$$\lambda = \begin{bmatrix} -3.33 & 0 & 0 & 0 \\ 0 & -0.52+0.46i & 0 & 0 \\ 0 & 0 & -0.52-0.46i & 0 \\ 0 & 0 & 0 & -1.47 \end{bmatrix}$$

The corresponding eigenvectors are,

$$Y = \begin{bmatrix} 0.297 & 0.295-0.107i & 0.295+0.107i & -0.316 \\ 0.551 & 0.323+0.506i & 0.323-0.506i & -0.943 \\ 0.505 & -0.374-0.018i & -0.374+0.018i & 0.050 \\ 0.594 & 0.633 & 0.633 & 0.094 \end{bmatrix}$$

The fast decaying eigen-coefficients will be w_1 and w_4. We do note that the separation between the eigenvalues is slightly less than what was seen in the BHR3 temporal shear case. This will cause the truncated solution to be less accurate, especially at early time. Figure 209 shows the truncated analytic flow map projected into the $k-S$ plane overlayed with full analytic trajectories.

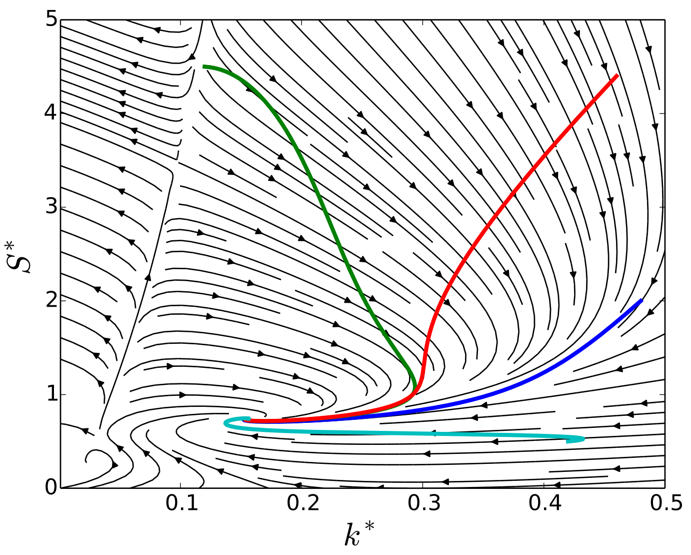


Figure 209: As the analytic trajectories are integrated forward in time they approach the plane on which the fast decaying eigen-coefficients are zero, causing the flow map to accurately describe the trajectories.

At early time there is considerable deviation between the full trajectories and the truncated flow map. This deviation is sensitive to initial conditions for b^* and a_y^*. As the profiles are integrated forward in time the fast decaying eigen-coefficients become negligible and the trajectories follow the flow map to the fixed point. For some initial conditions the fast decaying eigen-coefficients become negligible at a much slower rate, which reduces the usefulness of this method.

xRage Comparison

Rayleigh-Taylor simulations were ran in xRage. The following parameters were used for every case,

dx	Domain	C_a	C_b	C_C	C_{a1}	C_{r1}	C_{r2}	C_{r3}	C_{r4}	C_{b2}	C_4	C_S	C_1	C_3
0.03125 cm	40 cm	0.3	0.3	0.56	3.2	0.3	0.6	0.42	1.8	2.0	1.2	4.2	1.2	0.0

Table 12: Summary of coefficients for xRage Rayleigh-Taylor runs.

Runs were initialized by varying k and S to be,

	Run 1	Run 2	Run 3
k_0^*	1e-4	1e-5	1e-6
S_0^*	1e-2	1e-3	1e-4
Color (Figure 210)	Blue	Red	Green

Table 13: Summary of initializations for xRage Rayleigh-Taylor runs.

Figure 210 shows the xRage trajectories overlayed on the truncated flow map. Since there is not as much separation between the fast and slow decaying eigencoefficients, the truncated flow map is not accurate away from the fixed point. The deviations between the xRage results and the flow map away from the fixed point are due to this, opposed to the integral method being inaccurate. To demonstrate this, Figure 210 also shows analytic profiles with initial conditions that coincide with a point in the xRage runs. This point was chosen to be when the profiles were close to, but not at self-similarity.

Overall there is decent agreement between the xRage results and the integral method. As the profiles approach the fixed point they do begin to coincide with the truncated flow map. A comparison between the analytic profiles and xRage results also suggests good agreement. The same qualitative behavior is observed for all three cases. In particular we observe Run 3 (green) and note the same behavior as the trajectories fall onto the plane in which the fast decaying eigencoefficients are zero. Additionally, the xRage profiles do spiral in to the fixed point; although this can't be easily seen in Figure 210. This behavior is precisely what one would expect with complex eigenvalues.

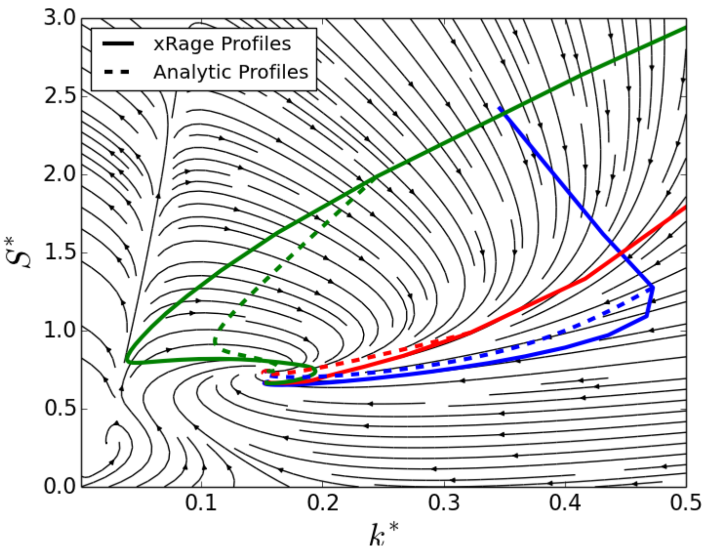


Figure 210: The xRage trajectories overlayed on the truncated flow map. The full analytic trajectories are the dashed lines.

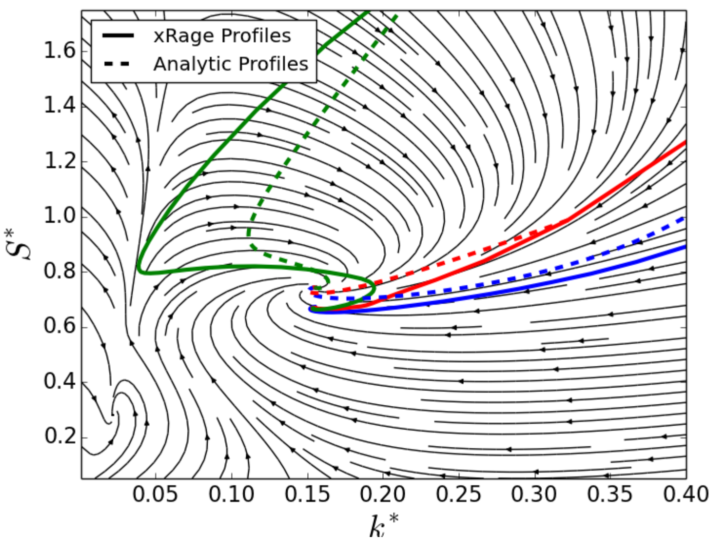


Figure 211: As in Figure 210, but zoomed in around the fixed point.

Conclusions

Good agreement is seen between integral method solutions and xRage results for both a temporal shear layer and Rayleigh Taylor layer with various turbulence models. For a temporal shear case with a BHR3 turbulence model, the eigenvalues were sufficiently separted to use an eigenspace decomposition to reconstruct a truncated system. Although the same concept was applied to Rayleigh Taylor cases, the eigenvalues were not as separated. Additionally, the more complex nature of Rayleigh Taylor increases the non-linearity of the system. This means we are visualizing the system by using a truncated solution of a linear approximation to a highly non-linear system. The errors in these approximations compound themselves, and the end result is that this method only accurately describes the system very close to the fixed point. This limited range of applicibility makes it difficult to visually compare xRage simulations and the integral method away from the fixed point in one 2D flow map. In order to visually compare a wider range of models, additional visualization techniques need to be considered. Overall, the integral method did compare well to xRage full-field simulations.

References

This chapter contains the references for all of the previous chapters.

References

[1] Gene M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, AFIPS '67 (Spring), pages 483–485, New York, NY, USA, 1967. ACM.

[2] Peter Amendt, SC Wilks, C Bellei, CK Li, and RD Petrasso. The potential role of electric fields and plasma barodiffusion on the inertial confinement fusion databasea). *Physics of Plasmas (1994-present)*, 18(5):056308, 2011.

[3] Kaushik Banerjee. *Kernel Density Estimator Methods for Monte Carlo Radiation Transport*. PhD thesis, University of Michigan, 2010.

[4] Didier Besnard, Francis H. Harlow, Rick M. Rauenzahn, and Charles Zemach. Turbulence Transport Equations for Variable-Density Turbulence and Their Relationship to Two-Field Models. LA-UR-12303. Technical report, Los Alamos National Laboratory, Los Alamos, NM, 1992.

[5] K. Burke. Perspective on density functional theory. *Journal of Chemical Physics*, 136:150901, 2012.

[6] D. Burton. Conservation of energy, momentum, and angular momentum in lagrangian staggered-grid hydrodynamics. Technical report, Lawrence Livermore National Laboratory, UCRL-JC-105926, 1990.

[7] Hoffmann C. Fundamental techniques for geometric and solid modeling. *Manufacturing and Automation Systems: Techniques and Technologies*, 48:347–356, 1991.

[8] E. Caramana, M. Shashkov, and Whalen P. Formulations of artificial viscosity for multi-dimensional shock wave computations. *J. Comput. Phys.*, 144:70–97, 1998.

[9] E.J. Caramana, D.E. Burton, M.J. Shashkov, and P.P. Whalen. The construction of compatible hydrodynamics algorithms utilizing conservation of total energy. *J. Comput. Phys.*, 146:227–262, 1998.

[10] J. Chihara. Interaction of photons with plasmas and liquid metals - photoabsorption and scattering. *J. Phys.: Condens. Matter*, 12(231), 2000.

[11] S.V. Coggeshall. Analytic solutions of hydrodynamics equations. *Physics of Fluids*, 3(5):757–769, 1991.

[12] S.V. Coggeshall. Group-invariant solutions og hydrodynamics. Technical report, Los Alamos National Laboratory, LA-UR-94-1277, 1994.

[13] S.V. Coggeshall and R.A. Axford. Lie group invariance properties of radiation hydrodynamics equations and their associated similarity solutions. *Physics of Fluids*, 29(8):2398–2420, 1986.

[14] J. Daligault. Half metal, half plasma: the physics of hot dense matter, 2014.

[15] E S Dodd, J A Baumgaertel, G R Mcnamara, J H Schmidt, and J H Cooley. A base-line model for ICF implosions in xRAGE A suite of base-line ICF capsule implosions is being set up for code validation at Los Alamos. In *44th Annual Anomalous Absorption Conference*. Los Alamos National Laboratory, 2014.

[16] E. S. Dodd, J. F. Benage, G. a. Kyrala, D. C. Wilson, F. J. Wysocki, W. Seka, V. Yu. Glebov, C. Stoeckl, and J. a. Frenje. The effects of laser absorption on direct-drive capsule experiments at OMEGA. *Physics of Plasmas*, 19(4):042703, 2012.

[17] ES Dodd, JH Schmidt, and JH Cooley. A base-line model for direct-drive icf implosions in the xrage code. *Bulletin of the American Physical Society*, 58, 2013.

[18] Thomas James Dolan. *Fusion Research*. 1982.

[19] G. Zerah F. Lambert, J. Clerouin. *Phys. Rev. E*, 82(036404), 2006.

[20] D. P. Flanagan and L. M. Taylor. An accurate numerical algorithm for stress integration with finite rotations. *Appl. Mech. Eng.*, 62:305–320, 1987.

[21] J.A. Fleck and J.D. Cummings. An implicit monte carlo scheme for calculating time and frequency dependent nonlinear thermal radiation transport. *Journal of Computational Physics*, 8:313–342, 1971.

[22] D. Frenkel and B. Smit. *Understanding Molecular Simulation*. Academic Press, Waltham, Massachusetts, second edition, 2006.

[23] S. H. Glenzer and R. Redmer. X-ray thomson scattering in high energy density plasmas. *Rev. Mod. Phys.*, 81(1625), 2009.

[24] Ronald Goldman. Intersection of lines in three-space. *Graphics Gems*, 1:304, 1990.

[25] IE Golovkin, JJ MacFarlane, PR Woodruff, LA Welser, DL McCrorey, RC Mancini, and JA Koch. Modeling of indirect-drive icf implosions using 1d hydrodynamic code with in-line collisional-radiative atomic kinetics. *Inertial Fusion Science and Applications 2003*, pages 166–169, 2003.

[26] Samuel K. Gutierrez. *libquo*, `https://github.com/losalamos/libquo`, 2013 (accessed July 3, 2014).

[27] J. P. Hansen and I. R. McDonald. *Theory of Simple Liquids*. Academic Press, Waltham, Massachusetts, third edition, 2006.

[28] Stefan (editor) Hiermaier. *Predictive Modeling of Dynamic Processes: A Tribute to Professor Klaus Thoma*. Springer Science and Business Media, LLC, 233 Spring Street, New York, NY, USA, 2009.

[29] JD Huba. Nrl plasma formulary 2009. Technical report, DTIC Document, 2009.

[30] Daniel M Israel. A Dynamical Systems Approach to the Alpha Problem for Rayleigh-Taylor. In *66th Annual Meeting of the APS Division of Fluid Dynamics*, Pittsburgh, PA, November 2013.

[31] Daniel M Israel. Private Communication. 2014.

[32] B. A. Jean, R. W. Douglass, G. R. McNamara, and F. A. Ortega. Dendritic meshing. Technical report, Los Alamos National Laboratory, 2011. LA-UR 11-04075.

[33] Y. S. Joe, A. M. Satan, and C. S. Kim. Classical analogy of fano resonances. *Phys. Scr.*, 74(259), 2006.

[34] V. V. Karasiev, T. Sjostrom, S. B J. Dufty, and Trickey. Accurate homogeneous electron gas exchange-correlation free energy for local spin-density calculations. *Phys. Rev. Lett.*, 112(076403), 2014.

[35] L.D. Landau and E.M. Lifshitz. *Fluid Mechanics*. Pergamon Press, Oxford, New York, 1987.

[36] R.J. LeVeque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press, The Pitt Building, Trumpington Street, Cambridge, United Kingdom, 2002.

[37] L. A. Collins M. P. Desjarlais, J. D. Kress. Electrical conductivity for warm, dense aluminum plasmas and liquids. *Physical Review Letters E*, 66(025401), 2002.

[38] Lawrence E. Malvern. Introduction to the mechanics of a continuous medium. pages 155–181, 1969.

[39] L. G. Margolin. A strain space framework for numerical hyperplasticity. *Math. Compu. Simul.*, 2012.

[40] L. G. Margolin and E. C. Flower. Numerical simulation of plasticity at high strain rate. *ASME Constitutive Modeling - Theory and Application*, pages 323–334, 1991.

[41] L.G. Margolin and D.E. Vaughan. Structure functions for finite volume shocks. Technical report, Los Alamos National Laboratory report, LA-UR-12-25922, 2012.

[42] A Marocchino, S Atzeni, and A Schiavi. Effects of non-local electron transport in one-dimensional and two-dimensional simulations of shock-ignited inertial confinement fusion targets. *Physics of Plasmas (1994-present)*, 21(1):012701, 2014.

[43] RJ Mason, RC Kirkpatrick, and RJ Faehl. Real viscosity effects in inertial confinement fusion target deuterium–tritium micro-implosions. *Physics of Plasmas (1994-present)*, 21(2):022705, 2014.

[44] J. P. Mithen. *Molecular dynamics simulations of the equilibrium dynamics of non-ideal plasmas*. PhD thesis, Trinity College, University of Oxford, 2012.

[45] Kim Molvig, Andrei Simakov, and Erik Vold. Classical transport equations for burning gas-metal plasmas. Technical report, Los Alamos National Laboratory, 2014.

[46] Nathaniel R. Morgan, Konstantin N. Lipnikov, Donald E. Burton, and Mark a. Kenamond. A Lagrangian staggered grid Godunov-like approach for hydrodynamics. *Journal of Computational Physics*, 259:568–597, February 2014.

[47] W.F. Noh. Errors for calculations of strong shocks using an artificial viscosity and an artificial heat flux. *J. Comput. Phys.*, 72:78–120, 1986.

[48] W.L. Oberkampf and T.G. Trucano. Verification and validation benchmarks. *Nuclear Engineering and Design*, 238:716–743, 2008.

[49] Ron Parker. Collisions in plasmas. chapter 3. 2006.

[50] Dean L. Preston, Davis L. Tonks, and Duane C. Wallace. Model of plastic deformation for extreme loading conditions. *J. Appl. Physics*, 93(1):211–220, 2003.

[51] T Rothman R. Rosner, D Hammer. Doe renew report on high energy density laboratory physics. 2009.

[52] PB Radha, TJB Collins, JA Delettrez, Y Elbaz, R Epstein, V Yu Glebov, VN Goncharov, RL Keck, JP Knauer, JA Marozas, et al. Multidimensional analysis of direct-drive, plastic-shell implosions on omegaa). *Physics of Plasmas (1994-present)*, 12(5):056307, 2005.

[53] G.M. Ramsey, S.D.and Hrbek. Semi-analytical investigation of strong converging shock waves. Technical report, Los Alamos National Laboratory, LA-UR-06-6739, 2006.

[54] S.D. Ramsey. Piston boundary conditions for lagrangian simulation of compressible flow similarity solutions. Technical report, Los Alamos National Laboratory, LA-UR-14-22318, 2014.

[55] J. Reisner, J. Bossert, and J. Winterkamp. *Numerical Simulations of Two Wildfire Events Using a Combined Modeling System (HIGRAD/BEHAVE)*. Dec 1997.

[56] Scott Runnels and Len Margolin. An integrated study of numerical shock shape, artifical viscosity, and plasticity. Technical report, Los Alamos National Labratory report, LA-UR-13-24226, 2013.

[57] Scott R. Runnels (editor). Final report from the 2012 computational physics student summer workshop. Technical report, Los Alamos National Laboratory, 2012.

[58] Scott R. Runnels (editor). Final report from the 2013 computational physics student summer workshop. Technical report, Los Alamos National Laboratory, 2013.

[59] H. R. Rüter and R. Redmer. *Ab initio* simulations for the ion-ion structure factor of warm dense aluminum. *Phys. Rev. Lett.*, 112(145007), 2014.

[60] G Schurtz, S Gary, S Hulin, C Chenais-Popovics, J-C Gauthier, F Thais, J Breil, F Durut, J-L Feugeas, P-H Maire, et al. Revisiting nonlocal electron-energy transport in inertial-fusion conditions. *Physical review letters*, 98(9):095002, 2007.

[61] John D Schwarzkopf, Daniel Livescu, Robert A Gore, Rick M. Rauenzahn, and J Raymond Ristorcelli. Application of a second-moment closure model to mixing processes involving multicomponent miscible fluids. *Journal of Turbulence*, 12(49), January 2011.

[62] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC, London, 1986.

[63] L. Silvestri, G. J. Kalman, Z. Donko, P. Hartmann, and H. Kaehlert. Fano-like resonances in strongly coupled binary coulomb systems, 2014.

[64] JM Soures, RL McCrory, CP Verdon, A Babushkin, RE Bahr, TR Boehly, R Boni, DK Bradley, DL Brown, RS Craxton, et al. Direct-drive laser-fusion experiments with the omega, 60-beam,¿ 40 kj, ultraviolet laser system. *Physics of Plasmas (1994-present)*, 3(5):2108–2112, 1996.

[65] A. N. Souza, D. J. Perkins, C. E. Starrett, D. Saumon, and S. B. Hansen. Predictions of x-ray scattering spectra for warm dense matter. *Phys. Rev. E*, 89(023108), 2014.

[66] Krista Stalsberg-Zarling and Rob Gore. The BHR2 Turbulence Model: Incompressible Isotropic Decay, Rayleigh-Taylor, Kelvin-Helmholtz and Homogeneous Variable-Density Turbulence. LA-UR-11-04773. Technical report, X Division: Theory and Design, Los Alamos National Laboratory, Los Alamos, NM, 2011.

[67] C. E. Starrett and D. Saumon. Electronic and ionic structures of warm and hot dense matter. *Phys. Rev. E*, 87(013104), 2013.

[68] C. E. Starrett and D. Saumon. A simple method for determining the ionic structure of warm dense matter. *High Energy Dens. Phys.*, 10(35), 2014.

[69] Matthias Teschner, Stefan Kimmerle, Bruno Heidelberger, Gabriel Zachmann, Laks Raghupathi, Arnulph Fuhrmann, M-P Cani, François Faure, Nadia Magnenat-Thalmann, Wolfgang Strasser, et al. Collision detection for deformable objects. In *Computer Graphics Forum*, volume 24, pages 61–81. Wiley Online Library, 2005.

[70] Erik Vold. Uniform DT 3T burn: Computations and sensitivities. Technical report, Los Alamos National Laboratory, 2010.

[71] T. G. White, S. Richardson, B. J. B. Crowley, L.K. Pattison, J. W. O. Harris, and G. Gregori. Orbital-free density-functional theory simulations of the dynamic structure factor of warm dense aluminum. *Phys. Rev. Lett.*, 111(175002), 2013.

[72] Allan B. Wollaber. *Advanced Monte Carlo Methods for Thermal Radiation Transport*. PhD thesis, University of Michigan, 2008.